LOS FICHEROS EN EL AMSTRAD CPC 464, CPC 664 Y CPC 6128

VICTOR J. CAMPO

ra-ma

Los Ficheros en el AMSTRAD CPC464, CPC664 y CPC6128

Víctor J. Campo

Licenciado en Ciencias Económicas

ra-ma

LOS FICHEROS EN EL AMSTRAD
CPC-464, 664 y 6128.

por Victor J. Campo.

(c) Copyright de la edición

RA-MA Editorial - 1987
Ctra. de Canillas, 144
28043 MADRID (España)
Telfs. 200 97 46/47

Impresion: Signo Impresores, S.A.
Albasanz nº 27
Madrid.

I.S.B.N.- 84-86381-23-1
D. Legal.- M-235-1.987

Reservados todos los derechos. No está permitida la reproducción parcial o total de este libro - sin el consentimiento por escrito del editor.

Agradecimiento

He de expresar mi agradecimiento a todas las personas que me han animado en la redacción de esta obra y, en especial, a Antonio Bellido y José Luis Ramírez, sin cuya ayuda no hubiera sido posible.

Dedicatoria

A Tere, Rubén y Noemí que con su comprensión y cariño me han animado en todo momento.

Introducción

Existen en la actualidad muchos libros dedicados al aprendizaje del lenguaje BASIC, algunos de ellos, explícitamente al Locomotive BASIC de Amstrad. Entre las materias tratadas suelen incluirse Las Matrices y Los Ficheros, pero quizá por la amplitud de estos temas, no con la necesaria extensión.

Por otro lado, mi experiencia dentro de la enseñanza me lleva a la conclusión de que dichos temas producen siempre en el estudiante una cierta dificultad de comprensión, de forma que aun dominando el lenguaje, se enfrentarán en sus aplicaciones ante un escollo que les impide profundizar en programación, quedando sus conocimientos informáticos mermados en tan importantes cuestiones.

Todo ello y la ausencia de publicaciones específicas, son razones que me han animado a escribir estas páginas, espero que su lectura y comprensión lleve a un mayor y más profundo conocimiento de los temas tratados.

El libro va dedicado expresamente a los usuarios de los CPC464, 664 y 6128 de Amstrad. En él se tratan temas de conocimiento general como matrices y ordenación, aplicables a cualquier ordenador y temas sobre ficheros, específicos de los citados modelos.

El contenido se ha dividido en 7 capítulos que, aun estando relacionados, permiten su lectura por separado. Forman tres grandes bloques: el primero (caps. 2 y 3) incluye todo lo relativo a la utilización de matrices (listas y tablas), como estructuras de almacenamiento transitorio y manipulación de datos. El segundo (caps. 4 y 5) está dedicado a los ficheros del Amstrad, tanto de acceso directo como secuencial, únicos posibles en los CPC; y el último bloque (caps. 6 y 7) pretende aportar una mayor utilidad a los programas EAMSWORD y ED.COM (CP/M), manejándolos desde BASIC y utilizándolos como eficaces creadores y manipuladores de ficheros, a la vez que se abren nuevas puertas para la utilización de ficheros creados desde diversos programas de aplicación, comerciales.

Además de las numerosas rutinas y ejemplos expuestos en cada capítulo, se incluyen como apéndices a los mismos, una serie de programas de utilidad, que además de su aportación como valor didáctico, permiten un uso directo como programas de aplicación.

El Autor

Indice

Capítulo 1 INTRODUCCION A LOS FICHEROS

Capítulo 2 MATRICES

Capítulo 3 ORDENACION

Capítulo 4 FICHEROS SECUENCIALES

Capítulo 5 FICHEROS DE ACCESO DIRECTO

Capítulo 6 FICHEROS CREADOS POR EAMSWORD

Capítulo 7 FICHEROS CREADOS POR ED.COM DE CP/M

CAPITULO 1

INTRODUCCION A LOS FICHEROS

1. **Definición de Fichero**

2. **Tipos de Fichero**
   * Secuenciales
   * Indexados
   * Acceso directo

3. **Concepto de Registro y Campo**

Introducción a los Ficheros

1. Definición de Fichero

Siempre resulta arriesgado dar una definición, por ello me limitaré a indicar que un **fichero** es un conjunto de datos relacionados entre sí, generalmente almacenado en un soporte magnético, disco o cinta. También cabe definirlo como conjunto de fichas o registros, por analogía con los ficheros o clasificadores de todos conocidos.

A cada fichero o conjunto de información se le asigna un nombre, por el que será reconocido, de forma que toda transmisión de datos entre la memoria del ordenador y el fichero, o viceversa, exige la referencia a dicho nombre.

2. Tipos de Fichero

Pueden realizarse diversas clasificaciones de ficheros, según su contenido, tipo de acceso, tiempo de existencia, etc.

Respecto a su contenido, los ficheros pueden clasificarse como:

Numéricos	(datos numéricos)
Alfanuméricos	(datos alfanuméricos, cadenas)
Mixtos	(datos numéricos y alfanuméricos)

Por otro lado, los datos, sea cual sea su naturaleza, pueden guardarse de distinta forma, lo cual da origen a una nueva clasificación:

Forma de almacenamiento de los datos	**Ficheros binarios** **Ficheros hexadecimales** **Ficheros octales** **Ficheros ASCII**

Los **ficheros binarios** son aquéllos cuya información es guardada en números binarios, es decir, en base 2; por tanto, dicha información se compone únicamente de ceros y unos, que son los dígitos disponibles en este sistema de numeración.

De igual forma, en los **ficheros hexadecimales y octales** la información está guardada en números de base 16 y 8, respectivamente.

Los **ficheros ASCII** son aquéllos cuya información es guardada de acuerdo con un código estándar cuyas siglas corresponden al "American Standard Code for Information Interchange" (Código Americano Normalizado para el Intercambio de Información). Esta norma permite la codificación de caracteres -un total de 128 signos- que puede ver en el manual de su ordenador.

De acuerdo con la forma en que se accede a los datos, los ficheros pueden ser:

* **Secuenciales**
* **Indexados**
* **Aleatorios o de acceso directo**

Ficheros secuenciales

Este tipo de ficheros va ligado íntimamente a la forma en que una cinta cassette guarda la información. Los datos son almacenados uno tras otro "secuencialmente" a medida que son introducidos.

El principal inconveniente de este método de almacenamiento, reside en que la lectura debe realizarse también secuencialmente; es decir, para leer un dato determinado, hay que leer antes todos los anteriores (como mínimo).

No todo son inconvenientes, su sencillez es un punto a su favor, así como la posibilidad de almacenar datos de muy diversa longitud, coexistiendo perfectamente en el fichero, cosa imposible en los de acceso directo.

Ventajas	Inconvenientes
Sencillez	Acceso no directo
Datos de cualquier longitud	Lento en la lectura de un dato
Disco y cinta	Obliga a trabajar con matrices grandes

Ficheros indexados

Este tipo de fichero no está disponible en los CPC de Amstrad, por lo que nos limitaremos a describir someramente sus características (sí son utilizables en la serie PCW).

Permiten el acceso directo a un dato determinado, a través de un parámetro **-índice-** que permite la búsqueda al ordenador a través de una tabla de índices. Su gran ventaja es que la búsqueda puede ser referida a un determinado campo dentro de un registro (ver estos conceptos).

Ficheros de acceso directo

Este tipo de ficheros no se encuentran implementados en el Locomotive BASIC de Amstrad, sino que deben ser utilizados a través de un programa de aplicación, incluido en la compra de los equipos (RANDOM FILES).

El acceso a un dato determinado es directo, entendiendo como dato un registro, no un campo. El ordenador, mediante un algoritmo, calcula la posición que ocupa en el disco el dato indicado, que lleva un parámetro para indicar el número de orden dentro del conjunto de datos.

Ventajas	Inconvenientes
Acceso directo a un registro	Sólo para discos
Menos ocupación de memoria principal	El acceso a un campo lo realiza el programador
No se necesitan matrices de almacenamiento transitorio	Datos de longitud fija
Proceso de datos, uno a uno	Desaprovecha espacio en el disco

Atendiendo al tiempo de existencia de los ficheros, pueden clasificarse, no exhaustivamente, en tres grandes grupos:

Transitorios (corto plazo)

Temporales (medio plazo)

Fijos (largo plazo)

Los **ficheros transitorios** son aquéllos cuya vida es muy corta. Generalmente duran sólo el tiempo en que están sujetos a modificación o creación, consolidándose posteriormente como temporales (ver como ejemplo los ficheros creados por ED.COM).

Los **temporales,** que son la inmensa mayoría, son ficheros dinámicos sujetos a continuas variaciones (actualización). Por último los **fijos,** son aquéllos que aun siendo susceptibles de modificaciones, éstas no son frecuentes, por lo que en un plazo medio pueden considerarse inalterables.

3. Concepto de Registro y Campo

Observando la figura que simula una ficha convencional de un fichero de alumnos, se desprende que para cada alumno existen una serie de epígrafes ubicados de una forma fija, que una vez completados constituyen los datos del alumno.

COLEGIO XXX

Alumno _____

Domicilio _____

Edad _____ Tel. _____

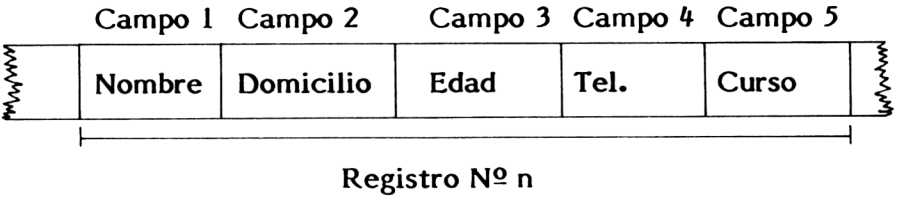
Curso _____

Pues bien, si por analogía, entendemos que un fichero físico es igual a uno informático, donde únicamente varía la forma de almacenar la información, el concepto de "ficha" se correspondería con el de **"registro"** y cada uno de los "epígrafes" de la ficha sería un **"campo".**

Fichero físico	Fichero informático	Tipo de información
Ficha	Registro	Varios datos
Epígrafe	Campo	Un solo dato

Es decir, volviendo al ejemplo, la ficha del alumno YYY sería un registro del fichero, mientas que el nombre del alumno será el primer campo de ese registro.

La información de una ficha quedaría almacenada en un fichero en la forma que indica la figura, puesto que ésta es depositada linealmente en el soporte de archivo (disco, cinta).



Un registro puede contener uno o más campos, al igual que un fichero puede contener uno o más registros, aunque lo habitual es, obviamente, ficheros conteniendo varios registros y a su vez, varios campos.

CAPITULO 2

MATRICES

1. **Conceptos previos**

2. **Matrices numéricas y alfanuméricas**

3. **Dimensiones de una matriz**
   * Ejercicios básicos sobre matrices

4. **Cargar una matriz**
   * Asignación con INPUT
   * Asignación con READ-DATA
   * Asignación desde ficheros

5. **Errores en el manejo de matrices**

6. **Operaciones más frecuentes con matrices**
   * Suma de los elementos de una lista
   * Suma de los elementos de una tabla
   * Búsqueda del mayor o menor de una lista
   * Corregir datos de una matriz
   * Anular un elemento en una lista
   * Añadir elementos a una lista
   * Búsqueda de elementos alfanuméricos de listas y tablas

7. **Ejercicios resueltos**
   1) Notas medias de un curso
   2) Producto de una matriz por un valor
   3) Inversa de una matriz
   4) Determinante de una matriz cuadrada
   5) Sistemas de ecuaciones lineales

8. **Relación de programas y rutinas utilizados en este capítulo**

Matrices

1. Conceptos previos

Generalidades

Las variables simples permiten tratar los datos uno a uno, pero en muchas ocasiones resulta conveniente tratar un conjunto de datos que tengan un nexo común y que permita que las manipulaciones de datos se realicen globalmente sobre dicho conjunto y no individualmente sobre cada variable simple.

De esta necesidad surgen las **variables indexadas,** conocidas como matrices, en inglés 'arrays' e incluso como **variables arreglo,** traducción macarrónica del inglés y excesivamente extendida en nuestro idioma.

El estudio de las matrices resulta fundamental en programación, ya que permite un cómodo, rápido y adecuado tratamiento de bloques de datos y ficheros de datos.

Concepto informático de matriz

Una **matriz** es un bloque o conjunto de celdas reservadas en la memoria del ordenador, que pueden ser referenciadas con un nombre simple y común a todos los elementos del conjunto, y distinguidas mediante un número (índice) de orden, que señala a una celda en particular.

Gráficamente estas estructuras nos recuerdan a las matrices matemáticas y de ahí su nombre, aunque su contenido y manejo se aparta del concepto puro matemático. Por ejemplo, un bloque formado por 5 celdas puede representarse de dos formas:

Horizontal

1	2	3	4	5

Matriz **fila** (vector)

Vertical

1
2
3
4
5

Matriz **columna** (vector)

Como veremos a continuación, también pueden reservarse bloques de dos dimensiones e incluso de más; por ejemplo:

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

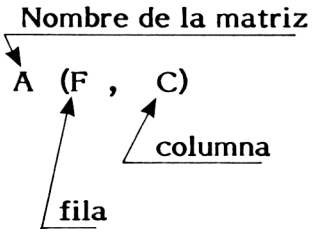
Los índices de una matriz de dos dimensiones, no corresponden a la numeración expuesta en el ejemplo anterior, sino que se utiliza un índice doble (2 números), el primer número indica la fila a que pertenece y el segundo, la columna. En el ejemplo anterior sería:

COLUMNAS

		1ª	2ª	3ª	4ª	5ª
FILAS	1ª	1.1	1.2	1.3	1.4	1.5
	2ª	2.1	2.2	2.3	2.4	2.5
	3ª	3.1	3.2	3.3	3.4	3.5

El elemento de índice 2.4, sería el situado en la fila 2ª y columna 4ª.

Por último, nos falta el nombre. Como indicamos anteriormente, el nombre es común para todo el conjunto, por tanto, si llamamos A al conjunto del ejemplo último, sus elementos serán conocidos como A11, A12, A13... A35, que es la nomenclatura matemática. En BASIC, los nombres de los elementos son A(1,1); A(1,2); A(1,3); .....; A(3,5); es decir, la sintaxis general es



Consideraciones

— El nombre de una matriz puede ser cualquiera que sea válido para variables, en el BASIC del Amstrad por ejemplo:

NOMBRES DE MATRICES

Válidos	No válidos
A	78 Z
HOLA	07
B7	4B

— Los **subíndices** comienzan su numeración en cero (0).

— El número de elementos de una lista (vector) es igual al índice del superior más uno.

A(0), A(1), A(2), A(3) 3+1=4 elementos

— Los subíndices, en su sentido más amplio, pueden ser:

- Números (4)
- Variable numérica (VEL)
- Expresión numérica (A*4/2)

— Tanto las variables numéricas como los resultados de las expresiones, contendrán o darán como resultado números enteros positivos (naturales).

— **Lista.-** Se denomina lista a una matriz de una sola dimensión (vector).

— **Tabla.-** Se conoce como tabla a las matrices de dos dimensiones.

— El número de elementos de una matriz de dos dimensiones es igual al producto del número de filas +1 por el número de columnas +1.

Así: A (F,C) donde F = 2 y C = 3, tendrá

(2+1) * (3+1) = 12 elementos

2. Matrices Numéricas y Alfanuméricas

Al igual que en las variables simples, existen matrices numéricas y alfanuméricas, dependiendo de su contenido:

TIPO	CONTENIDO	NOMBRE
Numéricas	Números	válido en BASIC
Alfanuméricas	Palabras	válido en BASIC seguido del símbolo $

Respecto al nombre, son válidas todas las reglas de uso en las variables simples. Debe hacerse notar que para el ordenador, las variables siguientes son distintas:

A 1 - Variable simple numérica
A 1 $ - Variable simple alfanumérica
A (1) - Elemento de la matriz numérica A
A $ (1) - Elemento de la matriz alfanumérica A$

Estas cuatro variables pueden coexistir perfectamente en un programa, por cuanto para el ordenador son totalmente distintas. De todos modos, no resulta aconsejable, pues la confusión viene por parte del programador y siendo tan amplia la posibilidad de la elección de nombres, el utilizar esta técnica resulta poco razonable, a menos que expresamente se desee programar con algo de ocultismo.

Como ejemplo de ambos tipos, representamos gráficamente el contenido de dos listas, una numérica y otra alfanumérica.

Nombre de la matriz	**Nº de elementos**	**Nombre y contenido de los elementos**			
DAT	4	DAT(0)	DAT(1)	DAT(2)	DAT(3)
		6	2.52	45	0
VINO$	3	VINO$(0)	VINO$(1)	VINO$(2)	
		RIOJA	VALDEPEÑAS	JEREZ	

Notas:

Las matrices numéricas sólo pueden contener números.

Las matrices alfanuméricas, además de textos o palabras propiamente dicho, pueden contener números, pero éstos serán considerados como palabras (conjunto de caracteres) y no como tales números.

Los elementos de las matrices numéricas están sujetos a las reglas y funciones matemáticas propias de los números y variables numéricas.

Los elementos de las matrices alfanuméricas únicamente podrán utilizarse con las reglas y funciones propias del BASIC para el manejo de palabras (**strings**).

A continuación exponemos un cuadro, no exhaustivo, de operaciones y funciones válidas y no válidas con elementos de matrices, a título de ejemplo:

Válido	No válido
A(1) * A(8)/2	A$(4)/2
A$(3) + Z$(0)	VAL (A(7))
X$(1) + "PEPE"	A$(0) + B(4) + 38
VAL (X$(1))	LEN (A(4))
LEFT$ (Z$,4)	MID$ (A(4), 6, 8)
A(I) * Z(K)	(I Y K VAR. NUM)

3. Dimensiones de una matriz

Hasta ahora hemos visto conceptos generales sobre las matrices, bajo el punto de vista informático y en concreto dentro del lenguaje BASIC.

Veamos ahora cómo se utilizan desde el BASIC del Amstrad (aunque únicamente nos referimos a los Amstrad CPC464, 664 y 6128, suele ser común a la mayoría de los ordenadores).

La sentencia DIM

La sentencia DIM se utiliza para establecer el número de elementos de una matriz, de tipo numérico o alfanumérico.

La sintaxis general es

DIM nombre (N)

siendo N el número de elementos menos uno (debido a que se empieza a contar en cero).

Así: **DIM A(4)** reserva 5 espacios de memoria para poder guardar en ellos los valores de los elementos de la matriz A, que serán A(0), A(1), A(2), A(3) y A(4).

En el caso de matrices con dos dimensiones, la sintaxis es:

DIM nombre (F,c)

siendo F el número de filas -1 y C el número de columnas -1.

Por ejemplo: **DIM A(2,3)** reservará (2+1) * (3+1) = 12 espacios de memoria, cuyos elementos se pueden representar gráficamente como:

A(0,0)	A(0,1)	A(0,2)	A(0,3)
A(1,0)	A(1,1)	A(1,2)	A(1,3)
A(2,0)	A(2,1)	A(2,2)	A(2,3)

La presencia del elemento cero, resulta a veces, para los programadores principiantes, perturbador, sobre todo en las matrices bidimensionales. Resulta práctica común hacer abstracción de todos los elementos cero y tratar la anterior matriz como si estuviese sólo compuesta por:

A(1,1)	A(1,2)	A(1,3)
A(2,1)	A(2,2)	A(2,3)

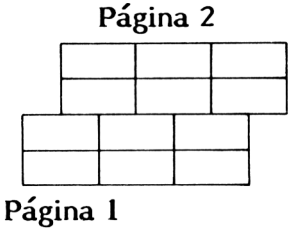
Aunque lícito, evidentemente se desaprovecha memoria. En los numerosos ejemplos que incluimos, veremos que a veces se utiliza esta técnica, aunque si queremos aprovechar al máximo la capacidad del ordenador, es conveniente utilizar todos los elementos disponibles. (En algunos ordenadores es posible seleccionar cuál será el primer elemento, el 0 o el 1, la orden más común es OPTION BASE 0 y OPTION BASE 1, respectivamente).

Todo lo visto hasta ahora respecto a DIM, es válido tanto para matrices numéricas como para las alfanuméricas.

Hasta el momento han aparecido matrices de una y dos dimensiones, pero en términos generales, pueden existir matrices de N dimensiones. Por ejemplo: una matriz tridimensional podía ser:

DIM A(2,3,2)

y estaría formada por 2 filas, 3 columnas y 2 páginas (sin contar los elementos 0, a fin de simplificar), esta matriz podía representarse gráficamente en la forma siguiente.



Página 1

A(1,1,1)	A(1,2,1)	A(1,3,1)
A(2,1,1)	A(2,2,1)	A(2,3,1)

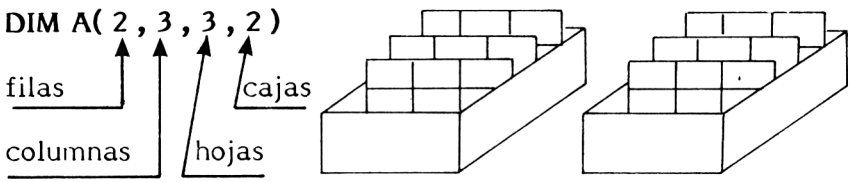
A(1,1,2)	A(1,2,2)	A(1,3,2)
A(2,1,2)	A(2,2,2)	A(2,3,2)

Página 2

En realidad y a la vista del gráfico, puede considerarse que una matriz de tres dimensiones, está compuesta por dos matrices bidimensionales, donde el último índice representa a la matriz a que pertenece el elemento.

Las matrices de cuatro dimensiones son algo más complicado de imaginar, y el grado de dificultad hace que sean poco operativas. Para mayores dimensiones 5, 6, etc., su estructura y manejo escapan bastante de la concepción humana, al menos de las personas comunes.

Una representación gráfica de las cuatro dimensiones, pudiera ser la de cajas, conteniendo hojas formadas por matrices bidimensionales.



Algún lector imaginativo puede hacer ejercicios mentales, considerando que las de cinco dimensiones son archivadores, llenos de cajas, que contienen fichas de tablas (matrices de 2 dimensiones). Y las de 6 habitaciones que contienen archivadores, que...

¿Hasta qué nivel se considera capaz de concebir? ¿Y de manejar? Esta última respuesta puede contestarla después.

Otras consideraciones

— En una misma sentencia DIM pueden agruparse varias matrices para su dimensionamiento.

La sintaxis general es:

DIM nombre 1 (dimensión), nombre 2 (dimensión), ...

Por ejemplo:

DIM A(4), A$(3,2), VIN (100,100)

— Si el índice de una matriz no va a superar el valor 10, no es necesario su previo dimensionamiento.

— La dimensión de una matriz está dada generalmente por números, pero también es válido dimensionar según el valor de una variable, o incluso, una expresión numérica.

Dimensión de una matriz { Número / Variable numérica / Expresión numérica

Por ejemplo:

1)
```
10 INPUT "Número de alumnos"; N
20 DIM AL$(N)
```

2)
```
10 INPUT "NUMERO DE FICHAS";NUM
20 DIM F$(NUM*2,4)
```

— Si la variable o la expresión no son números enteros, se toma como dimensión el entero más próximo, así:

```
10 N = 3.2
20 DIM A(N*2)
```

La dimensión de A será la parte entera de 3.2*2 = 6

— Si por alguna razón, una matriz no va a volver a utilizarse como tal y los valores contenidos no son de ningún interés, puede liberarse el espacio reservado en memoria mediante la orden ERASE.

La sintaxis general es:

ERASE nombre 1, nombre 2, ...

Donde **nombre** representa a las matrices a eliminar; por ejemplo:

10 DIM A(10), B$(5), Z(100)	Crea 3 matrices
.	
.	
.	
210 ERASE A, Z	Elimina las matrices A y Z, liberando el espacio ocupado por ellas.

Una vez eliminada una matriz, puede volver a utilizarse su nombre para crear una nueva; tarea que se denomina redimensionamiento y que sólo puede realizarse después de un ERASE.

Siguiendo con el ejemplo, sería válida una línea como la siguiente:

```
.
.
.
480 DIM A(24)
```

Ya que la matriz A fue borrada con anterioridad.

4. Cargar una matriz

Genéricamente, denominamos cargar una matriz a la operación por la cual asignamos a cada uno de sus elementos un valor.

Cuando se utiliza la orden DIM, automáticamente el ordenador asigna un cero a todos los elementos o una cadena vacía ("") en las alfanuméricas.

Por ejemplo:

DIM A(3)

A(1)	A(2)	A(3)	A(4)	Elementos
0	0	0	0	Asignación

DIM B$(2)

A$(0)	A$(1)	A$(2)
""	""	""

Las dobles comillas representan a una **cadena vacía,** es decir, una palabra formada por "ningún carácter", ni siquiera el blanco.

La tarea de carga o asignación viene regida por bucles FOR, encargados de la manipulación de los elementos de las matrices, aprovechándose la variable del FOR como variable índice de la matriz. Para más de una dimensión utilizaremos los bucles anillados, tantos como dimensiones tenga la matriz.

Asignación con LET

Dado que LET es en el BASIC la sentencia de asignación por antonomasia, nos referimos a ella en primer lugar.

La forma elemental será:

LET variable indexada = valor (numérico o alfanumérico)

Ejemplos:

1) Valores distintos

```
10 DIM A(2)
20 LET A(0) = 4
30 LET A(1) = 6
40 LET A(2) = 9
```

A(0)	A(1)	A(2)
0	0	0
4	6	9

2) Con los mismos valores para todos los elementos (inicializar)

```
10 DIM A$(3)
20 FOR I=0 TO 3
30 LET A$(I)="PEPE"
40 NEXT I
```

A$(0)	A$(1)	A$(2)	A$(3)
PEPE	PEPE	PEPE	PEPE

El uso de LET es muy variado y su potencia estriba en utilizar otras variables y no números en la asignación. También es útil cuando se desea inicializar una matriz con todos sus valores iguales, incluso poniéndolos a cero, pues debe tenerse en cuenta que la sentencia DIM sólo puede utilizarse una sola vez para la misma matriz; por tanto, si queremos restaurar en un momento dado las condiciones iniciales, necesitaremos un bucle como el del ejemplo 2, para la puesta a cero (o vacío) de la matriz.

Asignación con INPUT

En determinadas ocasiones necesitamos introducir los datos por teclado, para lo cual utilizamos la sentencia de entrada INPUT.

También es posible utilizar INKEY, INKEY$ e incluso el joystick, pero su aplicación es más restringida.

Veamos con un ejemplo, el uso de INPUT:

```
10 DIM A(4)
20 FOR I=0 TO 4
30 INPUT A(I)
40 NEXT I
```

En una asignación como la indicada, cuando el número de datos a introducir es elevado, es frecuente que se produzcan errores como consecuencia de la pérdida del orden en dicha entrada. Para evitarlos, es suficiente el introducir previamente al INPUT un PRINT que nos indique el número de orden del dato en curso. Como ejemplo, añada al programa anterior la línea:

```
25 PRINT "DATO ";I;
```

Haga nuevamente RUN y observe el efecto.

Asignación con READ-DATA

Cuando una matriz ha de contener, al menos inicialmente, los mismos valores tantas veces como se use el programa (valores constantes iniciales), resulta conveniente -si no son un número excesivo- almacenar dichos valores en líneas DATA y cargar la matriz con READ.

Veremos un ejemplo. Si deseamos tener en una matriz los días de la semana, de forma que al 1 le corresponda el lunes, al 2 el martes, etc... podemos utilizar el siguiente procedimiento:

```
10 DIM DS$(7)
20 FOR I=1 TO 7
30 READ DS$(I)
40 NEXT I
50 DATA LUNES, MARTES, MIERCOLES, JUEVES,
VIERNES, SABADO, DOMINGO
```

Se observa cómo el bucle lo establecemos desde 1 y no desde 0, a fin de hacer corresponder al lunes el índice 1, premisa del ejercicio. El elemento D$(0) existirá y contendrá el vacío (ningún carácter).

Los bucles de lectura, como el expuesto en éste y anteriores ejemplos, resulta conveniente agruparlos, de forma que desde el FOR al NEXT todo se encuentre en la misma línea, ya que son operaciones relacionadas entre sí.

Por ejemplo:

```
20 FOR I=1 TO 7 : READ DS$(I) : NEXT
```

Deliberadamente no incluimos la variable I del FOR, en la sentencia NEXT, pues en el Amstrad no es necesario, aunque a veces resulta más conveniente su uso por razones de claridad.

Asignación desde ficheros

Aunque este tema será tratado con amplitud, adelantamos que las matrices también pueden cargarse con datos contenidos en ficheros; para ello, se utiliza la sentencia INPUT, con indicación del canal del disco que en los Amstrad corresponde al #9.

Para poder extraer datos de un fichero, es necesario en primer lugar abrirlo; esto se realiza con la instrucción OPENIN que podemos traducir como abrir hacia dentro, o mejor, **abrir para leer.** La instrucción OPENIN contiene además el nombre del fichero, bien como tal entre comillas o mediante una variable alfanumérica que lo contenga.

```
        { "Nombre del fichero"
OPENIN  {
        { VAR $
```

Tras terminar la lectura, el fichero ha de ser cerrado mediante la orden CLOSEIN, sin indicar nombre.

Veamos todo ello con un sencillo ejemplo comentado.

```
10 INPUT "Nombre del fichero"; NOM $   Introducir el nombre
20 OPENIN NOM $                        Abrir el fichero
30 INPUT #9, N                         Leer el primer dato,
                                       que suponemos repre-
                                       senta el número total
                                       de datos que contiene
40 DIM A(N-1)                          Dimensionamiento en
                                       función de N
50 FOR K=0 TO N-1                      Bucle para leer N da-
                                       tos
60 INPUT #9, A(K)                      Lectura
70 NEXT K
80 CLOSEIN                             Cerrar archivo
```

Observe que si no se utiliza el elemento Ø de la matriz, las líneas 40 y 50 deben modificarse en la forma:

```
40 DIM A(N)
50 FOR K=1 TO N
```

Aprovechamos el ejemplo para indicar que puede ser conveniente guardar el valor de N (número de datos), precisamente en ese índice cero tan polémico.

En este caso, además de las variables de la 40 y 50, debemos utilizar una línea extra después del DIM y fuera del FOR, que haga:

```
A(0) = N
```

El número de línea podía ser 45, 75 o posterior a la 80, el resultado es idéntico.

5. Errores en el Manejo de Matrices

1) Subscript out of range

El subíndice utilizado es demasiado grande o demasiado pequeño. El límite inferior es 0 y el superior viene dado por la sentencia DIM correspondiente.

Ejemplo:

```
10 DIM A(8)
20 A(5) = 24
30 A(9) = 6
40 H = -2
50 A(H) = 4
```

Las líneas 30 y 50 dan error, puesto que -2 y 9 están fuera del rango de la matriz

2) Array already dimensioned

La matriz que trata de dimensionarse ya lo estaba anteriormente. No se puede redimensionar una matriz, ni aun con la misma dimensión.

Ejemplo:

```
10 DIM A(10)
.
.
.
180 DIM A(10)   o   180 DIM A(24)
```

Otros tipos de error pueden surgir, pero afectarán a las variables, sintaxis, límite de memoria, etc., pero no a las matrices propiamente.

6. Operaciones más frecuentes con Matrices

Suma de los elementos de una lista

En muchas ocasiones necesitamos obtener la suma de todos los elementos de una lista numérica (matriz de 1 dimensión).

Para ello nos valdremos como siempre de un bucle, cuya variable sirva como índice de los elementos de la matriz y de una variable simple que llamaremos SUMA.

El programa tendrá tres partes: la primera de carga, por cualquiera de los métodos expuestos, la segunda de suma, y la tercera, la salida por pantalla del resultado.

```
10 ´ CARGA DE DATOS
20 DIM A(6)
30 FOR K=0 TO 6
40 READ A(K)
50 NEXT K
60 ´ SUMA DE DATOS                    (M 1)
70 FOR K=0 TO 6
80 SUMA=SUMA+A(K)
90 NEXT K
100 ´ SALIDA DE RESULTADOS
110 CLS:PRINT "LA SUMA DE LOS NUMEROS"
120 FOR K=0 TO 6
130 PRINT A(K);"+";
140 NEXT K
150 PRINT "=";SUMA
200 DATA 4,3,6,12,45,8,2
```

Es interesante volver a insistir en la posibilidad de almacenar la suma en el elemento cero de la matriz, eliminando la variable simple SUMA.

Observe que como son 7 los datos, el nuevo dimensionamiento será precisamente DIM A(7), ya que el elemento A(0) queda reservado para la suma. Los límites de los bucles no serán 0 y 6, sino 1 y 7, respectivamente.

Veamos cómo quedaría el programa tras introducir las modificaciones apropiadas.

```
10 ´ CARGA DE DATOS
20 DIM A(7)
30 FOR K=0 TO 6
40 READ A(K)
50 NEXT K
60 ´ SUMA DE DATOS                    (M 2)
70 FOR K=1 TO 7
80 A(0)=A(0)+A(K)
90 NEXT K
100 ´ SALIDA DE RESULTADOS
110 CLS:PRINT "LA SUMA DE LOS NUMEROS"
120 FOR K=1 TO 7
130 PRINT A(K);"+";
140 NEXT K
150 PRINT "=";A(0)
200 DATA 4,3,6,12,45,8,2
```

Suma de los elementos de una tabla

Con las tablas (matrices de dos dimensiones) se presentan tres casos distintos:

* Suma de elementos por filas
* Suma de elementos por columnas
* Suma de todos los elementos

Veamos con un ejemplo los tres casos. Sea la tabla

					Σ FILAS	
	2	6	5	1	14	
	4	-3	2	0	3	
	3	5	1	-1	8	
Σ COLUMNAS	9	8	8	0	25	Σ TOTAL

El programa cargador, común a los tres cálculos, podía ser (supuestos los datos de la tabla contenidos en líneas DATA):

```
10 ´ CARGAR LA MATRIZ
15 CLS
20 DIM A(3,4)
30 FOR F=1 TO 3
40 FOR C=1 TO 4
50 READ A(F,C)
70 NEXT C
90 NEXT F
100 DATA 2,6,5,1,4,-3
110 DATA 2,0,3,5,1,-1
```

(M 3.1)

* Suma de todos los elementos

La suma de todos los elementos no reviste ningún problema, se establecen dos bucles iguales a los de lectura y en la variable simple TOTAL se van acumulando uno a uno todos los elementos.

```
200 ´ SUMA DE TODOS LOS ELEMENTOS
210 FOR F=1 TO 3
220 FOR C=1 TO 4
230 TOTAL=TOTAL+A(F,C)
240 NEXT C,F
```

(M 3.2)

Observe que la expresión: TOTAL = TOTAL + A(F,C) no debe interpretarse en sentido algebraico, puesto que ello equivaldría a decir que A(F,C)=0, ya que si despejamos en la expresión, tenemos TOTAL - TOTAL = A(F,C). Por tanto, el sentido es que el nuevo valor de la variable TOTAL ha de ser igual al valor antiguo más el valor del elemento A(F,C).

* Suma de filas

Dado que el número de filas es tres, necesitaríamos tres variables simples para almacenar la suma de los elementos de cada una de ellas y como consecuencia, tres bucles, uno por fila.

Pero observando las líneas 200 a 240, estructura de dos bucles, y considerando que el primero controla las filas, se deduce que sería más útil considerar una matriz unidimensional de tres elementos, cuyo índice viene controlado precisamente por el bucle de las filas.

Si llamamos SF (suma de filas) a la matriz, el programa sería:

```
300 ´ SUMA POR FILAS
310 FOR F=1 TO 3
320 FOR C=1 TO 4
330 SF(F)=SF(F)+A(F,C)
340 NEXT C,F
```

(M 3.3)

La matriz SF no se dimensiona previamente, puesto que sus elementos no superan el límite de 10 (ver Capítulo 2 - Otras consideraciones).

* Suma de columnas

La tercera parte del cálculo, suma de columnas, es igual a la anterior. También utilizamos una matriz sin previo dimensionamiento para almacenar las sumas de las cuatro columnas.

La diferencia con la anterior estriba en que los bucles están cambiados; primero el de columnas y luego el de filas.

```
400 ´SUMA POR COLUMNAS
410 FOR C=1 TO 4
420 FOR F=1 TO 3                    (M 3.4)
430 SC(C)=SC(C)+A(F,C)
440 NEXT F,C
```

Por último, procedemos a sacar por pantalla todos los datos en forma de tabla, tal como se planteó al principio.

```
500 ´ SALIDA DE RESULTADOS EN FORMA DE
510 FOR  F=1 TO 3
520 FOR C=1 TO 4
530 PRINT A(F,C);
540 NEXT C
550 PRINT "*";SF(F)                 (M 3.5)
560 NEXT F
570 PRINT "*****************"
580 FOR C=1 TO 4
590 PRINT SC(C);
600 NEXT C
610 PRINT "*";TOTAL
```

Nota: Para mayor claridad del ejercicio, se ha prescindido de los elementos de subíndice cero, práctica que será común en ulteriores ejemplos.

El programa ya está completo, haga RUN para comprobar y obtendrá una tabla con las sumas de las filas, de las columnas y la suma total, tal como se indica a continuación.

```
 2    6    5    1    *   14

 4   -3    2    0    *    3

 3    5    1   -1    *    8
*****************************
 9    8    8    0    *   25
```

Búsqueda del mayor o menor de una lista

* Números

Antes de entrar en el campo de la ordenación, como introducción, veamos cómo encontrar el mayor o el menor elemento de una lista o tabla.

Utilizando los datos del ejemplo M1 y las líneas 10 a 50 y la 200, que componen la carga de una lista en la matriz A, añadimos las siguientes líneas para la búsqueda del mayor.

```
10 ' CARGA DE DATOS
20 DIM A(6)
30 FOR K=0 TO 6
40 READ A(K)
50 NEXT K
200 DATA 4,3,6,12,45,8,2                    (M 4)
300 ' BUSQUEDA DEL MAYOR
310 FOR K=0 TO 6
320 IF A(K)>MAYOR THEN MAYOR=A(K)
330 NEXT K
340 PRINT "EL MAYOR DE LOS NUMEROS"
350 FOR K=0 TO 6
360 PRINT A(K);
370 NEXT K
380 PRINT "ES EL";MAYOR
```

La línea 310 es la encargada de ver quién es el mayor en cada vuelta del bucle, que maneja los siete elementos de la lista.

La variable MAYOR contiene el número mayor encontrado hasta una determinada vuelta del bucle.

Cuando K = 0 el valor de MAYOR = 0, por tanto en la primera vuelta MAYOR = A(0), si el siguiente valor es mayor se producirá un cambio, en caso contrario, continúa el mismo valor como mayor.

Veamos los valores que en el ejemplo toman las variables que intervienen:

Vuelta	K	A(K)	MAYOR
1	0	4	4
2	1	3	4
3	2	6	6
4	3	12	12
5	4	45	45
6	5	8	45
7	6	2	45

La búsqueda del menor es análoga, basta con cambiar el símbolo > de la línea 320 por el de <, y el nombre de la variable MAYOR por otro más apropiado.

* **Cadenas (palabras)**

El concepto de mayor y menor, aplicado a caracteres, es distinto del numérico.

Los caracteres están ordenados de menor a mayor en la forma alfabética, es decir A, B, C...., Z. Dicha ordenación no es arbitraria, sino que corresponde a un código estándar, que en el caso del Amstrad (y de la mayoría de los ordenadores), es el código ASCII.

En el apéndice se incluye una tabla completa de dicho código. Como se puede comprobar, a las letras mayúsculas del alfabeto les corresponde un valor que en números decimales comienza con el 65 para la A y finaliza con el 90 para la Z.

ASCII							
	Carácter	A	B	C	D	.....	Y	Z
	Valor decimal	65	66	67	68	.....	89	90

Ahora bien, observará que las letras minúsculas tienen valores distintos, y es que en realidad, son distintos caracteres para el ordenador. Veamos dichos valores:

ASCII							
	Carácter	a	b	c	d	.....	y	z
	Valor decimal	97	98	99	100	.....	121	122

Una letra (o carácter) es mayor que otra, si su valor ASCII lo es.

C > A ya que 67 > 65

Ahora bien, ¿qué ocurre si comparamos **A** mayúscula y **a** minúscula?

Dado que **a** vale 97 y **A** vale 65, resulta que:

a > A

La conclusión inmediata es que si establecemos comparaciones, debemos hacerlo con letras de un mismo tipo, mayúsculas o minúsculas, pero no con ambas.

Practique con el siguiente programa y vea los resultados.

```
10 INPUT "Introduzca una letra ";L1$
20 INPUT "Introduzca otra letra";L2$
30 IF L1$>L2$ THEN PRINT L1$;" es mayor que ";L2$:GOTO 50
40 PRINT L2$;" es mayor que ";L1$
50 PRINT "------------------------"
60 GOTO 10
```

(M 5)

¿Qué ocurre con las cadenas? Veámoslo de forma práctica. Utilice el programa anterior modificando (si lo desea) el texto de los INPUT, sustituyendo LETRA por PALABRA y observe los resultados obtenidos introduciendo palabras en mayúsculas en minúsculas o mixtas.

En la tabla siguiente incluimos algunos ejemplos que nos permitirán sacar consecuencias para nuestros propósitos.

	PALABRA (A)	PALABRA (B)	RESULTADO
1	TOLEDO	Toledo	A < B
2	ZAPATO	abanico	A < B
3	PERA	PERO	A < B
4	124	125	A < B
5	cara	caRa	A > B

1. **TOLEDO** es menor que **Toledo,** puesto que las letras minúsculas tienen un valor ASCII mayor.

2. **ZAPATO** es menor que **abanico** por igual motivo, a pesar de que **zapato** comienza con la última letra del abecedario y **abanico** con la primera.

3. Puesto que las tres primeras letras son iguales, la decisión de ser **PERO** > **PERA,** es debido únicamente a la letra **O** frente a la **A.**

4. Los números tratados como caracteres también responden al concepto matemático de mayor o menor.

5. **cara** es mayor que **caRa** puesto que **r** es mayor que **R,** en términos ASCII.

Hagamos ahora un programa que nos permite hallar la cadena menor de una lista o, lo que es lo mismo, la cadena que alfabéticamente es la primera.

```
10 ´ MENOR DE UNA LISTA ALFANUMERICA (Primero alfabeticamente)
20 DIM A(9)
30 FOR K=0 TO 9
40 READ A$(K)
50 NEXT K
60 ´ Busqueda
70 MENOR$="Z"
80 FOR K=0 TO 9
90 IF A$(K)<MENOR$ THEN MENOR$=A$(K)
100 NEXT K
110 CLS:PRINT "LA MENOR DE LAS PALABRAS"
120 FOR K=0 TO 9:PRINT A$(K):NEXT K
130 PRINT "ES:  ";CHR$(24);MENOR$;CHR$(24)
140 DATA PERA,COCHE,RIO,ZAPATO,RELOJ
150 DATA ARBOL,CEPA,MESA,FLOR,LAPIZ
```

(M 6)

La variable auxiliar MENOR$, ha de inicializarse con un valor alto (letras altas) con el fin de asegurar un buen funcionamiento del programa, puesto que en caso contrario, la variable contendría -por omisión- la cadena vacía, que es evidentemente un valor más bajo que cualquier palabra.

Por lo demás, el programa es similar al utilizado anteriormente para los números.

Corregir datos de una matriz

Cuando el número de datos es elevado, la probabilidad de cometer errores en la entrada por teclado es alta.

Para prevenir estos errores, necesitamos dos herramientas: la primera, una **entrada de datos clara,** sobre todo en el caso de tablas, de forma que en pantalla se señale perfectamente la fila y columna en curso. La segunda, un **programa depurador** que permita corregir los datos erróneos antes de pasar a la fase de procesado de los mismos.

Veamos con un ejemplo cada una de las partes:

Sea una tabla numérica de 5 x 7 datos, donde las filas representan "conceptos de gasto" y las columnas "los días de la semana".

La mejor forma de identificación es, evidentemente, llamar a las cosas por su nombre, de forma que utilizaremos dos matrices auxiliares (alfanuméricas), para almacenar los nombres de los "conceptos" y de los "días" de la semana. Por ser constantes podemos utilizar líneas DATA.

Las matrices serán:

Conceptos	C$ (5)
Días	D$ (7)
Datos	D (5,7)

```
10 ´ RUTINA PARA CORRECCION DE DATOS
20 DIM C$(5),D$(7),D(5,7)
30 ´ CARGAR CONCEPTOS
40 FOR K=1 TO 5:READ C$(K):NEXT
50 ´ CARGAR DIAS
60 FOR K=1 TO 7:READ D$(K):NEXT
70 DATA Alimentacion,Vestido,Transporte,Casa,Varios
80 DATA Lunes,Martes,Miercoles,Jueves,Viernes,Sabado,Domingo
100 ´ ENTRADA DE DATOS
110 CLS
120 FOR F=1 TO 5
130 PRINT "Gastos ";C$(F);" correspondientes al "
140 FOR c=1 TO 7
150 PRINT D$(C);
160 INPUT D(F,C)
170 NEXT C
180 PRINT "----------------------------------------"
190 NEXT F
```

(M 7)

Como puede ver, tras hacer RUN la entrada de datos resulta clara, se indica en pantalla el tipo de gastos y después el día de la semana. De esta forma, conseguimos una introducción ordenada de los datos, no obstante, a pesar de la claridad, el introducir números induce a errores de teclado, así que vamos a añadir la herramienta de depuración necesaria.

```
200 ´ DEPURACION DE DATOS
210 CLS:PRINT "DATOS ACTUALES"
220 PRINT "==============="
230 FOR F=1 TO 5
240 FOR C=1 TO 7
250 PRINT D(F,C);
260 NEXT C
```

(M 8)

```
270 PRINT
280 NEXT F
290 INPUT "ALGUN DATO ERRONEO (S/N) ";Z$
300 Z$=UPPER$(Z$)
310 IF Z$="N" THEN 500
320 IF Z$="S" THEN 340
330 GOTO 290
340 INPUT "CUANTOS ERRORES ";E
350 FOR K=1 TO E
360 INPUT "INTRODUZCA LA FILA Y LA COLUMNA ";F,C
370 PRINT "VALOR ERRONEO=";D(F,C),
380 INPUT "NUEVO VALOR ";D(F,C)
390 NEXT K
400 GOTO 200
500 ' PROCESADO DE LOS DATOS
```

Las líneas 210 a 280 se encargan de mostrar los datos en forma de tabla, tal y como se introdujeron por el teclado. Si la variable Z$ contiene una **s** por respuesta afirmativa al INPUT, el programa pregunta (340) cuántos errores se han detectado. El bucle que se inicia en 350 pide que se introduzca la fila y la columna del elemento erróneo, muestra después su valor y pide el nuevo.

La línea 400 devuelve el control al principio de la depuración, con el fin de que se muestre la nueva tabla con las correcciones efectuadas, permitiendo volver a corregir o, en caso de estar bien, con contestar **N** al INPUT de la línea 290, se abandonará la rutina de depuración para continuar el programa (línea 500 en adelante).

Evidentemente, el ejemplo expuesto es simplemente una idea básica susceptible de muchas mejoras. La entrada de datos y la depuración o filtro de los mismos, es una técnica que algunos programadores han convertido en "arte".

La rutina anterior es, como hemos dicho, básica, pero suficiente en la mayoría de los casos, aunque el lector puede y debe modificarla a su manera. Piense que cada programador tiene un "sello" personal de hacer las cosas, incluso un cierto estándar de representación en pantalla.

Anular un elemento de una matriz

Una tabla o una lista está compuesta de elementos generalmente dinámicos, es decir, sujetos a continuas variaciones del contenido de los mismos.

Anular uno de los elementos es una tarea frecuente. Imaginemos una lista como la siguiente:

1	6	7	8	5	9

compuesta de 6 elementos, en la que deseamos eliminar el cuarto elemento de valor 8, es decir, que la nueva lista tendrá 5 elementos en vez de 6.

La rutina de eliminación podía ser algo así como la que exponemos a continuación:

```
10 ´ RUTINA DE ELIMINACION
20 ´ Carga de la lista
30 N=6: ´No. de elementos
40 DIM A(N)
50 FOR K=1 TO N:READ A(K):NEXT
60 DATA 1,6;7,8,5,9
70 ´ Eliminacion de un elemento
80 INPUT "Que elemento quiere eliminar ";EL
90 FOR K=EL TO N-1
100 A(K)=A(K+1)
110 NEXT K
120 A(N)=0
130 ´Imprimir nueva tabla
140 FOR K=1 TO N:PRINT A(K):NEXT K
```

(M 9)

Una vez conocida la posición del elemento a eliminar, por el INPUT de la línea 80.

Establecemos un bucle cuyos límites son: la posición a eliminar y el número de elementos de la lista menos uno. En realidad, lo que hacemos es un intercambio, asignando al elemento A(K) el valor del siguiente: A(K+1).

En el ejemplo propuesto, al llegar a la línea 110 la lista en memoria será:

1	6	7	5	9	9

El elemento cuarto que era un 8, ha pasado a ser el siguiente (el 5), y lo mismo ocurre hasta el penúltimo (el N-1). Mas ¿qué ocurre con el último? Este no ha variado, así que el 9 ocupa la 5ª posición por el intercambio, y la 6ª por origen. Si no se desea que se repita, sino simplemente que la lista se acorte, hacemos el último elemento igual a cero, como puede verse en el listado, en la línea 120.

La línea 140 nos muestra la nueva tabla.

Como habrá podido adivinar, esta operación de anulación es una de las opciones que debe ofrecer un programa de base de datos, como veremos en los apartados correspondientes.

Queda todavía por ver algo que quizá el lector ya haya echado en falta. Todo lo expuesto está bien, pero ¿qué ocurre si no conocemos la posición del elemento a eliminar, sino su valor? Esto se da mucho en el caso de textos, donde se conoce la palabra a eliminar, pero no su posición y sobre todo, cuando las listas son largas y en las tablas, más difícil de detectar la posición, puesto que ésta viene dada por dos coordenadas: fila y columna.

Este caso presenta dos problemas, la **búsqueda del elemento a eliminar** y la **eliminación.** La última parte es idéntica al caso anterior, teniendo en cuenta que la eliminación del último -que antes era poner a cero- ahora será poner en cadena vacía.

Veamos cómo quedaría el ejemplo anterior aplicado a este caso.

```
10 ´ RUTINA DE ELIMINACION (textos)
20 ´ Carga de la lista
30 N=6: ´No. de elementos                    (M 10)
40 DIM A$(N)
50 FOR K=1 TO N:READ A$(K):NEXT
60 DATA platano,pera,manzana,uva,melon,naranja
70 ´ Eliminacion de un elemento
80 INPUT "Nombre del elemento a eliminar ";EL$
90 FOR K=EL TO N
100 IF A$(K)=EL$ THEN EL=K:GOTO 120
110 NEXT K
120 FOR K=EL TO N-1
130 A$(K)=A$(K+1)
140 NEXT K
150 A$(N)=""
160 ´Imprimir nueva tabla
170 FOR K=1 TO N:PRINT A$(K):NEXT K
```

Hay que tener muy en cuenta que el nombre a eliminar debe ser introducido con la misma sintaxis que figura en las DATA, habida cuenta de lo explicado anteriormente sobre mayúsculas y minúsculas.

El bucle 90-110 es el encargado de encontrar el nombre en el momento que se verifica la igualdad de la línea 100, el valor de K indicará precisamente la posición en la lista, por lo que la variable EL recoge dicho valor.

Las líneas 120 a 140 constituyen el bucle de actualización que se inicia en el valor EL (posición de la palabra a borrar) y finaliza en N=1, al igual que en el programa anterior.

Por último, la línea 150 hace que el último elemento no se repita y desaparezca, acortando la lista en un elemento (el que fue suprimido).

Añadir elementos a una matriz

Este caso es justo el contrario del anterior. En principio, trataremos de introducir un nuevo elemento en una lista, bajo el supuesto de que la matriz es lo suficientemente grande para ello.

La inserción cabe hacerla al final de la lista o en una posición intermedia cualquiera.

El hacer una inserción en cola no supone ningún problema, basta saber el número de elementos cargados que tiene y hacer el siguiente igual al valor a añadir.

Si la posición es intermedia habrá que hacer un hueco y correr -desde ese lugar- un puesto a la derecha al resto de los elementos.

Veamos esto gráficamente con un ejemplo y elaboremos una rutina que haga estas funciones. Sea la lista:

N=5

2	5	8	7	6		

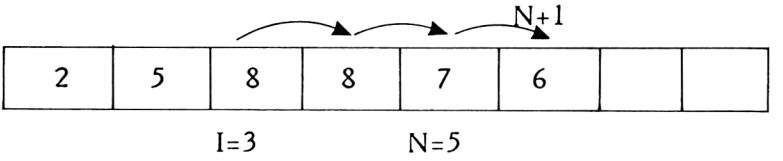
que dispone de 8 elementos, de los cuales los 5 primeros contienen valores y los 3 últimos vacíos (serían 0 por la orden DIM).

Ahora deseamos intercalar un 1 en tercer lugar; es decir, entre el 5 y el 8 de la lista.

```
10 ´ INSERCION EN UNA LISTA
20 ´ Carga de datos originales
30 N=5:DIM A(N+3)
40 FOR K=1 TO N:READ A(K):NEXT
50 DATA 2,5,8,7,6                    (M 11)
60 ´ Insercion
70 INPUT "POSICION DE INSERCION ";I
80 INPUT "VALOR ";V
90 FOR K=N+1 TO I STEP -1
100 A(K)=A(K-1)
110 NEXT K
120 A(I)=V
130 ´Salida de la nueva lista
140 FOR K=1 TO N+1:PRINT A(K):NEXT
```

Como puede ver, el bucle de inserción se ha planteado a la inversa que en los ejemplos anteriores. Desde un valor mayor en una unidad al número de elementos cargados, hasta la posición I de inserción, puesto que N+1 > I, el valor del paso será -1.

Cuando finalice el bucle, la lista tendrá la siguiente forma:



Por último, sólo falta introducir en la posición I el nuevo valor, que en el ejemplo hemos supuesto 1. De ello se encarga la línea 120.

La tabla quedará definitivamente en la forma:

2	5	1	8	7	6		

Como podrá comprobar al hacer RUN en el programa anterior.

La inserción del ejemplo ha supuesto una rotación de elementos a la derecha de la lista. Por supuesto, puede realizarse a la izquierda, basta con modificar los límites del bucle, comenzando en la posición anterior a la primera; es decir, en la cero y finalizando en I-1, y haciendo que A(K) = A(K+1).

Para listar todos los elementos, el bucle de la 140 tendrá como límite inferior cero en vez de uno.

```
10 ´ INSERCION EN UNA LISTA (Por la izquierda)
20 ´ Carga de datos originales
30 N=5:DIM A(N+3)
40 FOR K=1 TO N:READ A(K):NEXT
50 DATA 2,5,8,7,6
60 ´ Insercion
70 INPUT "POSICION DE INSERCION ";I
80 INPUT "VALOR ";V
90 FOR K=0 TO I-1
100 A(K)=A(K+1)
110 NEXT K
120 A(I)=V
130 ´Salida de la nueva lista
140 FOR K=0 TO N+1:PRINT A(K):NEXT
```

(M 12)

Otro problema que abordaremos más adelante, será la inserción de cadenas, en una lista ordenada alfabéticamente, de forma que la nueva cadena respete la ordenación. En el capítulo siguiente trataremos todo lo relativo a la ordenación.

Búsqueda de elementos alfanuméricos en listas y tablas

En el apartado tercero de este capítulo vimos cómo **encontrar el mayor o menor de una lista,** tanto numérica como alfanumérica. Ahora bien, la búsqueda de textos debe enfocarse en un contexto más amplio que el numérico. En este sentido y como complemento, estudiaremos a continuación la búsqueda de un elemento por igualdad o por contenido.

Búsqueda por igualdad

En una lista de nombres, por ejemplo, es frecuente tratar de buscar uno determinado y una vez localizado, listarlo junto con atributos inherentes al propio nombre. Veamos un ejemplo.

Imaginemos una breve agenda telefónica cuyo contenido sea: nombre, apellidos y teléfonos. En principio, cabría la posibilidad de utilizar 3 matrices de una dimensión, tales como:

N$ Para nombres
A$ Para apellidos
T$ Para teléfonos

La N$ sería la matriz principal sobre la que vamos a efectuar la búsqueda de nombres y A$ y T$ serían las matrices de atributos. A un determinado nombre, elemento N$(i), le corresponderían los atributos A$(i) y T$(i) para los apellidos y teléfonos, respectivamente.

Si mantenemos esta estructura matricial, la rutina de búsqueda podía ser:

```
1000 ' RUTINA DE BUSQUEDA POR IGUALDAD
1010 INPUT "NOMBRE A BUSCAR ";NOM$
1020 FOR K=1 TO N
1030 IF N$(K)=NOM$ THEN 1100
1040 NEXT K
1050 PRINT "NO EXISTE ";NOM$
1060 GOTO 1130
1100 PRINT N$(K)
1110 PRINT A$(K)
1120 PRINT T$(K)
1130 RETURN
```

(M 13)

Mediante la línea 1010 introducimos el nombre que buscamos, teniendo en cuenta que dicho nombre debe introducirse tal y como figura en la lista, es decir, respetando el tipo de letra, mayúsculas y minúsculas, tal como se comentó con anterioridad.

La línea 1020 establece un bucle extendido a todos los elementos de la lista (N elementos).

En 1030 se efectúa la comparación con el **símbolo de igualdad** -título de este epígrafe. Debe tenerse en cuenta que la igualdad ha de ser absoluta, carácter a carácter. Si dicha igualdad se verifica es que se ha encontrado el elemento buscado, y por tanto, pasamos a la línea 1100 y siguientes, que imprimen el nombre, los apellidos y el teléfono, aprovechando el índice K del bucle, común a nombre y atributos.

Si se agotase el bucle, sería una señal inequívoca de que ha recorrido toda la lista sin encontrar el nombre buscado; por ello, la 1050 nos avisa de este hecho y produce un retorno al programa principal.

Otra forma de enfocar el problema, más correcta, es utilizar una sola matriz de tres dimensiones. Si llamamos N$ a esta matriz, los elementos de la fila iésima representarían a:

N$(i,1)	Nombre
N$(i,2)	Apellidos
N$(i,3)	Teléfono

Suponiendo que -por mayor claridad- no utilizamos los elementos cero de la tabla, como hemos hecho en muchos ejemplos anteriores e incluso en la rutina expuesta.

La confección de una rutina como la anterior, para este tipo de estructura matricial, no presenta ningún problema. Basémonos en la anterior.

```
1000 ' RUTINA DE BUSQUEDA POR IGUALDAD
1010 INPUT "NOMBRE A BUSCAR ";NOM$
1020 FOR K=1 TO N
1030 IF N$(K,1)=NOM$ THEN 1100
1040 NEXT K
1050 PRINT "NO EXISTE ";NOM$
1060 GOTO 1130                    (M 14)
1100 PRINT N$(K,1)
1110 PRINT N$(K,2)
1120 PRINT N$(K,3)
1130 RETURN
```

El sistema expuesto a través de los ejemplos de búsqueda por igualdad, presenta varios problemas, entre ellos dos muy importantes. Por un lado, la entrada de datos para cargar las matrices ha de ser meticulosa, no vale introducir por ejemplo: **PEPE** y luego **Antonio**, hay que tomar un criterio fijo y, además, hay que tener cuidado con los espacios; a fin de cuentas, también son caracteres. Las funciones UPPER$ y LOWER$ nos serán útiles a estos efectos.

Por otro lado, lo más lógico en el ejemplo expuesto es introducir el nombre seguido de los apellidos, o apellidos seguidos del nombre, según los gustos o el destino del programa general. En este caso, para buscar un elemento, debemos recordar su nombre y dos apellidos escritos correctamente, cosa ya algo difícil, sobre todo si se trata de una agenda de regular dimensiones.

Algunos de estos inconvenientes pueden solucionarse utilizando la técnica de **búsqueda por inclusión,** que exponemos a continuación.

Búsqueda por inclusión

El Basic del Amstrad dispone de una función de búsqueda dentro de cadenas que nos facilitará el trabajo, aunque puede realizarse sin ella.

Esta función es INSTR que lleva 3 parámetros; el primero de ellos opcional.

INSTR (partida, cadena en la que busca, cadena buscada)

El primer parámetro indica la posición a partir de la cual se va a efectuar la búsqueda. Si no se le indica, toma 1 por defecto.

El segundo, la cadena en la cual va a buscar.

El tercero, la subcadena buscada.

Si la subcadena buscada existe, devuelve la posición del carácter a partir del cual se ha encontrado la coincidencia. Si no existe, devuelve un cero (0).

Veamos un ejemplo. Sea la cadena B$="Las noches de invierno son frías", en la cual se busca la subcadena A$="invierno".

INSTR (1,B$,A$) o INSTR (B$,A$)

Nos devolverá el valor 16, que es la posición ocupada por la "i".

La aplicación en una búsqueda se realizará con un IF, comprobando que INSTR da un valor distinto de cero, lo cual indica que la subcadena está contenida.

Pruebe el ejemplo expuesto, tecleando las siguientes instrucciones sin número de línea:

```
B$="Las noches de invierno son frías"
A$="invierno"
PRINT INSTR (B$,A$)
```

Cambie ahora A$ por otro texto y compruebe los resultados. Le sugerimos los siguientes valores o contenidos de A$: "L", "de", "no", "frías". Como verá, localiza cualquier subcadena, sea o no palabra propiamente dicha.

Pues bien, basándonos en las facilidades de esta función, obtengamos una rutina de búsqueda (por inclusión).

Utilicemos el ejemplo del apartado anterior (agenda), empleando una matriz más amplia que contenga:

Nombre y apellidos	N$(i,1)
Dirección	N$(i,2)
Ciudad	N$(i,3)
Provincia	N$(i,4)
Teléfono	N$(i,5)

Lo cual implicaría un dimensionamiento DIM N$(N,5), si despreciamos los elementos cero.

La rutina podía ser:

```
1000 RUTINA DE BUSQUEDA POR INCLUSION
1010 INPUT "NOMBRE A BUSCAR ";NOM$
1020 FOR K=1 TO N
1030 IF INSTR (N$(K,1),NOM$)<>0 THEN GOSUB 1100
1040 NEXT K
1050 GOTO 1150                    (M 15)
1100 FOR J=1 TO 5
1110 PRINT N$(K,J)
1120 NEXT J
1130 PRINT "PRESIONE UNA TECLA PARA CONTINUAR"
1140 CALL &BB18
1150 RETURN
```

Ahora al introducir el nombre a buscar, podemos hacerlo de muchas formas, por ejemplo:

- Nombre
- Primer apellido
- Segundo apellido
- Inicial del nombre
- etc.

Hemos incluido además una pequeña mejora, consistente en que a pesar de que haya encontrado la subcadena, nos permita seguir buscando, por si hubiese más nombres o subcadenas que cumplan la misma condición.

Es fácil que en una agenda existan dos o más Pepes, etc., y no digamos si lo que buscamos son iniciales.

Podía mejorarse más, en el sentido de que si el nombre o subcadena buscada es la deseada, se controlase el continuar la búsqueda o salir, mediante un INPUT e IF en consonancia.

Algo así como:

```
1130 INPUT "DESEA CONTINUAR LA BUSQUEDA (S/N)"; Z$
1140 IF Z$="S" THEN RETURN
1150 IF Z$="N" THEN CON=1:GOTO 1035
1160 GOTO 1130
```

(M 16)

siendo la 1035

```
1035 IF CON=1 THEN 1050
```

En el apéndice al capítulo 4, puede ver un programa que permite llevar una agenda, en el cual se utiliza la técnica de búsqueda por inclusión.

Búsqueda en listas ordenadas

En los dos apartados anteriores nos hemos referido, aunque de forma implícita, a listas no ordenadas, donde la localización de un elemento ha de realizarse, "mirando" desde el primero al último, pues no existe ningún otro criterio de búsqueda.

Ahora bien, si una lista está ordenada alfabéticamente y el elemento buscado comienza por **z**, evidentemente se encontrará por el final. Luego en listas ordenadas, podemos establecer alguna regla o algoritmo de búsqueda, conducente a conseguir una mayor velocidad en la misma.

Uno de estos métodos es el de BUSQUEDA BINARIA. Su fundamento resulta sencillo: si comparamos el elemento buscado con el central de la lista y resulta menor, es que estará por encima, por el contrario, estará por debajo. Veámoslo gráficamente:

A5

K0	=	índice del primer elemento
Kc	=	índice del elemento central
Kn	=	índice del último elemento

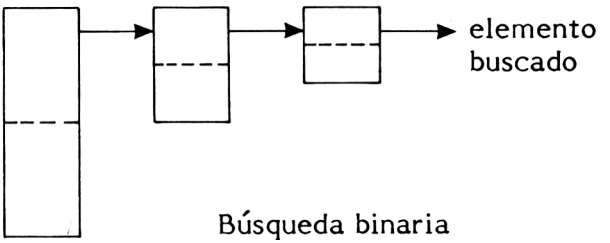
Si la cadena buscada es B$ y resulta

B$ A$(Kc) —sí→ B$, está en la primera mitad

↓ no

B$, está en la segunda mitad

Con una sola partición podemos desechar la mitad de los elementos, de manera que si realizamos una nueva partición, comparando con el elemento central de la mitad válida, podemos quedarnos con un cuarto de los elementos para seguir la búsqueda. De esta forma la confluencia es muy rápida.



Búsqueda binaria
Particiones sucesivas a la mitad

Basándonos en este sencillo método, podemos elaborar una rutina rápida de búsqueda de cadenas alfanuméricas en listas ordenadas. Ha de tenerse en cuenta que cada partición supone calcular los nuevos límites de la lista y el elemento central, que corresponde al valor medio de los límites. Dichos elementos singulares vienen representados por las variables K0, Kn y Kc, respectivamente.

```
10 ' CARGA DE LA LISTA ORDENADA
15 ' ..........      ..............
20 N=20:DIM A$(N)
30 FOR K=1 TO N
40 READ A$(K)
50 NEXT K
60 DATA AREA,AXILA,BALA,BARCO,CEBRA,CORCHO,DATIL,DEDO,DINAMO,DOLOR
70 DATA FELIZ,FUGA,GATO,HUECO,INDIO,JAMON,KILO,LIMON,MANGA,PERRO
100 ' BUSQUEDA BINARIA
110 INPUT "NOMBRE BUSCADO";NOM$
120 K0=1:KN=N
130 KC=INT((KN-K0)/2)+K0
140 IF NOM$=A$(KC) THEN 200
150 IF NOM$>A$(KC) THEN K0=KC:GOTO 130
160 KN=KC
170 GOTO 130
200 PRINT "El nombre ";NOM$;" ocupa el lugar ";KC
```

(M 17)

Las líneas 10 a 70 son auxiliares y su misión es comprobar el funcionamiento de la rutina.

Como origen, se toma en la línea 120 los valores 1 y N para los límites; el valor medio correspondiente se calcula en 130.

La línea 140 controla la búsqueda por igualdad, aunque se podía haber adoptado el método de la inclusión.

La línea 150 sirve de báscula para la partición. Observe cómo el nuevo valor de K0, en el caso de que el nombre buscado esté por debajo, se hace igual al valor central y en caso contrario, el límite inferior Kn, se hace igual al central.

Pruebe esta rutina y comprobará su perfecto funcionamiento, pero introduzca un nombre no contenido en las DATA. Ahora el programa buscará sin descanso y sin detenerse y, evidentemente, no la encontrará, así que debemos añadir un control para poder salir de este bucle infinito.

¿Qué condición pondremos? Introduzca la siguiente línea para ver qué pasa con los valores de límites y centro:

```
135 PRINT K0, KC, KN
```

Haga un nuevo RUN, introduzca palabras no existentes en las DATAS y observará que en la quinta o sexta vuelta, los 3 límites son iguales. Como el número de vuelta depende evidentemente del número de palabras de la lista, no nos sirve como indicador, así que tomamos la observación de las igualdades como conclusión, bastándonos estrictamente que sean iguales los dos primeros.

Añadamos las siguientes líneas y ahora sí funcionará con cualquier palabra, exista o no en la lista.

```
135 IF K0=Kc THEN 220
210 END
220 PRINT "No existe"; nom$
```

7. Ejercicios Resueltos

Ejercicio 1: Dadas las calificaciones obtenidas por los alumnos de una clase, por asignatura, obtener la nota media de cada alumno, la nota media de cada asignatura, el número de alumnos aprobados y suspensos y el porcentaje de ambos.

Datos: Vamos a utilizar como ejemplo sólo 5 alumnos, aunque el programa es válido para cualquier número de alumnos. Asimismo, supondremos 6 asignaturas, valor que puede también variarse.

Nombre del Alumno	ASIGNATURA					
	Mat	Soc	Dib	Len	Rel	Quim
Antonio García	6	7	8	7	9	9
Luis Conde	8	9	10	9	10	9
José Moreno	4	5	3	5	5	2
Rubén López	3	2	6	5	5	4
Luis Díaz	7	6	5	4	6	5

Dadas las premisas del ejercicio, utilizaremos 3 matrices

CONTENIDO	DIMENSION	TIPO
Nombre de los alumnos	(5)	alfanumérica
Asignaturas	(6)	alfanumérica
Notas	(5,6)	numérica

La nota media de cada alumno será la suma de los elementos de la matriz por filas, dividido por 6; mientras que la media de cada asignatura será la suma por columnas, dividido por 5 (alumnos).

La operación de sumar filas y columnas de una tabla fue explicada en el segundo apartado de este capítulo, en cuyo ejercicio nos apoyaremos.

Los datos de las matrices se cargarán desde el teclado, por medio de sentencias INPUT. Más adelante veremos la forma de almacenar los datos en ficheros.

```
10 ´ N O T A S    M E D I A S
20 MODE 1
30 INPUT "NUMERO DE ALUMNOS";AL
40 INPUT "NUMERO DE ASIGNATURAS";AS
50 ´DIMENSIONAMIENTO Y ENTRADA DE DATOS
60 DIM AL$(AL),AS$(AS),N(AL,AS)
70 FOR K=1 TO AL
80 PRINT "NOMBRE DEL ALUMNO";K
90 INPUT AL$(K)
100 NEXT K
110 FOR K=1 TO AS
120 PRINT "ASIGNATURA";K
130 INPUT AS$(K)
140 NEXT K
150 CLS
160 FOR F=1 TO AL
170 PRINT "----------------------------------------"
180 PRINT "NOTAS DEL ALUMNO: ";AL$(F)
190 FOR C=1 TO AS
200 PRINT AS$(C);
210 INPUT N(F,C)
220 NEXT C,F
230 ´ CALCULO DE LAS MEDIAS
235 ´ Por alumno
236 PRINT "NOTAS MEDIAS DE LOS ALUMNOS"
237 Z$=STRING$(40,"="):PRINT Z$
250 FOR F=1 TO AL
260 FOR C=1 TO AS
270 N(F,0)=N(F,0)+N(F,C)
280 NEXT C
290 PRINT "NOTA MEDIA DE ";AL$(F);"  "; USING "RR.R";N(F,0)/AS
300 NEXT F
310 ´ Por asignatura
315 PRINT :PRINT "NOTAS MEDIAS DE LAS ASIGNATURAS"
316 PRINT Z$
320 FOR C=1 TO AS
330 FOR F=1 TO AL
340 N(0,C)=N(0,C)+N(F,C)
350 NEXT F
360 PRINT "NOTA MEDIA DE ";AS$(C);"  "; USING "RR.R";N(0,C)/AL
370 NEXT C
400 ´ ALUMNOS APROBADOS Y SUSPENDIDOS
410 FOR F=1 TO AL
420 FOR C=1 TO AS
430 IF N(F,C)<5 THEN SUS=1
440 NEXT C
450 SUSPENSOS=SUSPENSOS+SUS
460 NEXT F
470 APROBADOS=AL-SUSPENSOS
480 PRINT :PRINT "APROBRADOS=";APROBADOS,"%=";  USING "RRRR.RR";APROBADOS/AL*100
490 PRINT :PRINT "SUSPENSOS =";SUSPENSOS,"%=";  USING "RRRR.RR";SUSPENSOS/AL*100
```

(M 18)

Ejercicio 2: Hallar el producto de una matriz por un escalar. El producto de una matriz (A) por un escalar K, es otra matriz (B), tal que todos sus elementos se obtienen B(i,j)=A(i,j)*K.

A continuación exponemos la rutina para calcular la matriz (B). La entrada y salida de resultados la dejaremos a criterio del lector. En los ejercicios anteriores hay varios ejemplos que sin duda le ayudarán.

```
10 ' PRODUCTO DE UNA MATRIZ POR UN ESCALAR
20 ' AGREGAR LA RUTINA DE CARGA
30 ' ..........  MATRIZ A(M,N)
40 ' ..........  ESCALAR=K
50 ' ..........
200 ' MATRIZ PRODUCTO
210 FOR F=1 TO M                    (M 19)
220 FOR C=1 TO N
230 B(F,C)=A(F,C)*K
240 NEXT C,F
300 ' AGREGAR RUTINA DE SALIDA DE RESULTADOS
310 ' .........
```

Ejercicio 3: Hallar la inversa de una matriz.

Este ejercicio y los dos siguientes van destinados al profesional y estudiante que utiliza y realiza ejercicios en los que es necesario el cálculo de la matriz inversa. Por todo ello, pasamos a escribir el programa remitiendo a un libro de Matemáticas a aquellos lectores interesados en el proceso de cálculo.

Como comprobación del programa, puede utilizar los siguientes ejemplos:

$$A=\begin{pmatrix} -1 & 6 \\ 4 & 3 \end{pmatrix} \text{Inversa } B=\begin{pmatrix} -0.1111 & 0.222 \\ 0.1481 & 0.037 \end{pmatrix} \qquad B=A\uparrow -1$$

$$A=\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \qquad \text{No tiene inversa (matriz singular)}$$

Nota: Si desea obtener más decimales, cambie el PRINT USING de la línea 560.

Si la matriz es grande, conviene que trabaje en MODE 2. Puede incluir esta orden en sustitución del CLS de la línea 20.

```
10 ' CALCULO DE LA MATRIZ INVERSA
12 ' ......      ..................
20 CLS:INPUT "ORDEN DE LA MATRIZ ";N
30 DIM A(N,N),B(N,N)
40 ' Entrada de datos de la matriz A
50 FOR F=1 TO N
55 PRINT "F I L A ";F
60 FOR C=1 TO N
70 PRINT "COLUMNA ";C;
80 INPUT A(F,C)
90 NEXT C
100 PRINT "----------------------------------------"
110 NEXT F
200 ' Validez de (A) ..................
220 FOR F=1 TO N
230 B(F,F)=1
240 NEXT F
250 FOR C=1 TO N
260 FOR F=C TO N
270 IF A(F,C)<>0 THEN 300
280 NEXT F
290 PRINT CHR$(24);"LA MATRIZ (A) ES SINGULAR Y POR TANTO NO TIENE INVERSA";CHR$
(24):END
300 ' Calculo de la matriz inversa (B)
310 FOR k=1 TO n
320 AUX=A(C,K)
330 A(C,K)=A(F,K)
340 A(F,K)=AUX
350 AUXB=B(C,K)
360 B(F,K)=AUXB
370 NEXT K
380 Z=1/A(C,C)
390 FOR K=1 TO N
400 A(C,K)=A(C,K)*Z
410 B(C,K)=B(C,K)*Z
420 NEXT K
430 FOR I=1 TO N
440 IF I=C THEN 500
450 Z=-A(i,c)
460 FOR J=1 TO N
470 A(I,J)=A(I,J)+A(C,J)*Z
480 B(I,J)=B(I,J)+B(C,J)*Z
490 NEXT J
500 NEXT I
510 NEXT C
520 ' Salida de resultados
530 PRINT "M A T R I Z    I N V E R S A"
540 FOR F=1 TO N
550 FOR C=1 TO N
560 PRINT  USING "<<<<<.<<<<";B(F,C);
570 NEXT C
580 PRINT
590 NEXT F
```

(M 20)

Ejercicio 4: Determinante de una matriz cuadrada.

Este programa permite calcular el valor del determinante de una matriz cuadrada (número de filas = número de columnas).

El procedimiento de cálculo utilizado es el de triangulación de la matriz. Este método se basa en que el producto de los elementos de la diagonal de la matriz triangulada es el valor del determinante.

Si se desea, puede incorporarse a la entrada de datos una rutina de depuración. Para ello, puede adaptarse la rutina del cuarto apartado (líneas 200 a 400). Los valores numéricos 5 y 7 de dicha rutina deben eliminarse y dejarlos como N (orden del determinante).

```
10 ' CALCULO DEL DETERMINANTE DE UNA MATRIZ CUADRADA
20 ' ......      ..................
30 CLS:INPUT "ORDEN DEL DETERMINANTE ";N
40 DIM A(N,N),Z(N)
50 ' Entrada de datos de la matriz
60 FOR F=1 TO N
70 PRINT "F I L A ";F
80 FOR C=1 TO N
90 PRINT "COLUMNA ";C;
100 INPUT A(F,C)
110 NEXT C
120 PRINT "--------------------------------------"
130 NEXT F
500 ' Validez del determinante (matriz no singular)
510 FOR C=1 TO N
520 FOR F=C TO N
530 IF A(F,C)<>0 THEN 560
540 NEXT F
550 PRINT CHR$(24);"LA MATRIZ  ES SINGULAR Y POR TANTO EL DETERMINANTE NULO";CHR
$(24):END
560 ' Calculo de la matriz inversa (B)
570 FOR k=1 TO n
580 AUX=A(C,K)
590 A(C,K)=A(F,K)
600 A(F,K)=AUX
610 NEXT K
620 Z(C)=A(C,C)
630 Z=1/A(C,C)
640 FOR K=1 TO N
650 A(C,K)=A(C,K)*Z
660 NEXT K
670 FOR I=1 TO N
680 IF I=C THEN 730
690 Z=-A(I,C)
700 FOR J=1 TO N
710 A(I,J)=A(I,J)+A(C,J)*Z
720 NEXT J
730 NEXT I
740 NEXT C
750 ' Calculo de los productos de la diagonal
760 P=1
770 FOR K=1 TO N:P=P*Z(K):NEXT K
780 ' Salida de resultados
790 PRINT :PRINT "El valor del determinante es ";P
```

(M 21)

Ejercicio 5: Resolución de sistemas de ecuaciones lineales.

El programa que acompañamos permite obtener las soluciones de un sistema de ecuaciones lineales con el mismo número de ecuaciones que incógnitas, siempre que el determinante de la matriz de coeficientes no sea nulo.

Se ha incluido una rutina de depuración de los datos de entrada, similar a la expuesta en el cuarto apartado.

Si se desea, pueden eliminarse las líneas 40 a 80, que son exclusivamente de presentación de la forma general de un sistema de ecuaciones lineales.

```
10 ' RESOLUCION DE SISTEMAS DE ECUACIONES
20 ' ..........    ..................
30 MODE 2
40  PRINT "Este programa permite calcular un sistema de  <n> ecuaciones con <n>
incognitas de la forma"
50 LOCATE 10,10: PRINT CHR$(24);" a11. X1 + a12. X2 + a13.X3 + a1n.Xn = b1  "
60 LOCATE 10,11: PRINT " a21. X1 + a22. X2 + a23.X3 + a2n.Xn = b2  "
70 LOCATE 10,12:PRINT " . . . . . . . . . . . . . . . . . . . . . . . . . . . . "
80 LOCATE 10,13: PRINT " an1. X1 + an2. X2 + an3.X3 + ann.Xn = bn  ";CHR$(24)
90 LOCATE 12,20:INPUT "INTRODUZCA EL NUMERO DE ECUACIONES";EC
100 DIM A(EC,EC+1)
110 ' Entrada de datos ..................................................
120 CLS
130 FOR F=1 TO EC
140 PRINT "INTRODUZCA LOS COEFICIENTES DE LA ECUACION";F
150 FOR C=1 TO EC
160  PRINT "A";F;C;"= ";LOCATE 10,C+1:INPUT "",A(F,C)
170 NEXT C
180 INPUT "TERMINO INDEPENDIENTE ";A(F,EC+1)
190 CLS
200 NEXT F
210 'sistema en pantalla ...............................................
220 CLS
230 FOR F=1 TO EC
240 FOR c=1 TO EC
250 PRINT A(F,C);"  ",
260 NEXT C
270 PRINT ":";A(F,EC+1)
280 NEXT F
290  PRINT :INPUT "DATOS CORRECTOS (S/N)     ";Z$
300 IF Z$="S" OR Z$="s" THEN 360
310 INPUT "CUANTOS DATOS INCORRECTOS";IN
320 FOR K= 1 TO IN
330 INPUT "INTRODUZCIR Fila,Columna,Valor correcto";F,C,A(F,C)
340 NEXT K
350 GOTO 210
360 ' Calculo ..........................................................
370 FOR F=1 TO EC
380 FOR C=F TO EC
390 IF A(F,C)<>0 THEN 410 ELSE NEXT C
400 PRINT CHR$(24);"   EL SISTEMA TIENE INFINITAS SOLUCIONES    ";CHR$(24):END
410 FOR K=1 TO EC+1
420 X=A(F,K):A(F,K)=A(C,K):A(C,K)=X
430 NEXT K
440 Y=1/A(F,F)
450 FOR K=1 TO EC+1:A(F,K)=A(F,K)*Y: NEXT K
460 FOR C=1 TO EC
470 IF C=F THEN 520
480 Y=-A(C,F)
490 FOR K=1 TO EC+1
500 A(C,K)=A(C,K)+A(F,K)*Y
510 NEXT K
520 NEXT C
530 NEXT F
540 'Salida de resultados ...............................................
550 CLS: PRINT "   S O L U C I O N    D E L    S I S T E M A"
560      PRINT "=============================================="
570 FOR C=1 TO EC
580 PRINT "X(";C;") =";  USING "<<<<<.<<<";A(C,EC+1)
590 NEXT C
```

(M 22)

CAPITULO 3

ORDENACION

1. Introducción
2. Ordenación por intercambio
3. Método BUBBLE SORT o de la burbuja
4. BUBBLE SORT con marca
5. Intercambio retardado
6. Búsqueda del mínimo y máximo. Doble pasada
7. Método de SHELL-METZNER
8. QUICKSORT
9. Apéndice: Programa general de ordenación

Ordenación

1. Introducción

Dentro de la gestión de datos, la obtención de listas ordenadas de los mismos, ocupan un lugar clave. Por ello, los programas de tratamiento de datos incorporan una o varias rutinas de clasificación, tanto para datos numéricos como alfanuméricos.

Las **rutinas de clasificación u ordenación,** conocidas como SORTs son, por tanto, independientes de los propios programas a que acompañan y secundarias, en el sentido de que el tiempo invertido por ellas debe ser mínimo, a fin de ceder el control lo más rápidamente al programa principal.

Los **tipos de ordenación** posibles dependen de la naturaleza de los datos y en términos generales pueden clasificarse de la forma siguiente:

Datos numéricos	De menor a manor
	De mayor a menor
Datos alfanuméricos	Alfabéticamente

Por supuesto que existen otros tipos de ordenación: por grupos alfabéticos (v.g. cadenas que comienzan por la misma letra); por marcas (v.g. números comprendidos entre 1.000 y 2.000); por atributos comunes (v.g. personas con la misma nacionalidad), etc.

Pero si variados son los tipos de ordenación que pueden establecerse, aún lo son más los métodos existentes para conseguirlo, desde los más sencillos e intuitivos, hasta sofisticados sistemas basados en algoritmos matemáticos complejos.

Esta diversidad de métodos no es de ningún modo caprichosa, sino que obedece a necesidades muy variadas. No es lo mismo tratar un pequeño número de datos que uno grande, ni tampoco resulta lo mismo ordenar una lista poco desordenada, que una de carácter caótico o aleatorio.

Por todo ello, no puede hablarse de un método mejor o peor que otro, si no va unido este juicio a dos parámetros: tamaño del grupo de datos y características del mismo.

En los epígrafes que siguen se exponen algunos de estas técnicas, indicándose la idoneidad de su aplicación según los parámetros citados.

Todas las rutinas de ordenación expuestas se complementan con las incluidas al final del capítulo y que se citan a continuación:

1ª **Entrada de datos.** Esta rutina permite la carga de datos que componen la lista desde el teclado, tanto numéricas como alfanuméricas.

2ª **Grabación de los datos en fichero.** Los datos introducidos por teclado son salvados en un fichero secuencial, en cinta o disco, según el modelo de Amstrad que utilicemos.

3ª **Lectura de ficheros.** Permite recuperar datos archivados.

4ª **Salida de datos ordenados.** Presenta los datos una vez ordenados por cualquiera de los métodos que se exponen.

Estas cuatro rutinas, así como las de ordenación, se han elaborado de forma que sean compatibles entre sí, lo cual permite que sean ensambladas mediante MERGE, constituyendo un completo programa de ordenación.

PGM.ORDENACION = R.1+R.2+R.3+R.ORDENACION+R.4

Para mayor comodidad del lector, se incluye un programa con las cuatro rutinas ensambladas y dirigidas desde un menú que permite además la carga (MERGE) de la rutina de ordenación deseada, trabajando todo conjuntamente.

2. Ordenación por intercambio

Este método de ordenación, también conocido como **dicotómico,** es quizá el más intuitivo. Se basa en recorrer la lista de datos desde el principio, comparando elementos contiguos e intercambiándolos si no están ordenados.

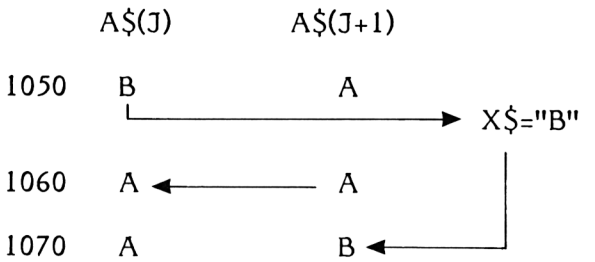
En una lista genérica A$(), se compara A$(1) con A$(2) y se coloca como A$(2) el mayor de los dos, luego se compara A$(2) con A$(3) y así sucesivamente.

Cuando se termina de recorrer la lista, tendremos el mayor situado en último lugar, es decir, en su sitio. Por tanto, en la segunda vuelta (2º bucle) ya no es necesario llegar hasta él, sino hasta el antepenúltimo y en general hasta el anterior al último elemento ordenado. Es decir, si I es la variable del primer bucle, el límite del segundo será N-I, siendo N el número de datos a ordenar.

Veamos cómo será una rutina que utilice este método.

```
1000 ' ORDENACION POR INTERCAMBIO
1015 ' ..........ORDENA1 ........
1020 FOR I=1 TO N-1
1030 FOR J=1 TO N-I
1040 IF A$(J)<=A$(J+1) THEN 1080
1050 X$=A$(J)
1060 A$(J)=A$(J+1)
1070 A$(J+1)=X$
1080 NEXT J
1090 NEXT I
```

Suponiendo que el elemento A$(J)="B" y A$(J+1)="A"



El método resulta sencillo y es quizá su mayor virtud; en cambio, su velocidad resulta baja. Hay que tener en cuenta que el número de vueltas es siempre fijo, de manera que si una lista está casi ordenada, tardará lo mismo que en otra aleatoria.

Otro tanto a su favor, aunque es común a otros métodos, es que no consume memoria, puesto que únicamente se utiliza una variable, la auxiliar, dado que la ordenación se realiza sobre la propia lista.

3. Método del BUBBLE SORT o de la burbuja

Las comparaciones se realizan comenzando por el primer elemento frente a todos los demás. Si en una comparación resulta ser menor, conserva su sitio, en caso contrario se intercambia.

En cada pasada el menor queda, pues, situado arriba, por ello recibe el nombre de **método de burbuja,** ya que los números ascienden por la lista como las burbujas en una bebida gaseosa. Por cierto, a estas alturas de la lectura del libro, quizá sea el momento de prepararnos un refresco a la vez que utilizamos el **bubble sort.**

Como puede observarse, en cada pasada se está seleccionando el mínimo de la serie, por lo que también se le conoce como **método de selección del mínimo.** Evidentemente, si se desea una ordenación en sentido inverso, el método es válido, basta con seleccionar el máximo en vez del mínimo, ello equivale a sustituir el operador de relación ">" de la línea 1040 por el "<", en la rutina que incluimos a continuación.

```
1000 ' ORDENACION METODO DE LA BURBUJA
1015 ' ........... ORDENA2 ............
1020 FOR I=1 TO N-1
1030 FOR J=I+1 TO N
1040 IF A$(I)>A$(J) THEN X$=A$(I):A$(I)=
A$(J):A$(J)=X$
1050 NEXT J
1060 NEXT I
```

Este método es muy similar al anterior. Sólo utiliza una variable auxiliar, de forma que los intercambios se realizan dentro de la propia matriz, por lo que el consumo de memoria es mínimo.

4. BUBBLE SORT con marca

Se trata de una variante del método anterior, que introduce una mejora en el tiempo de procesado.

En el anterior, si la lista a tratar está compuesta de elementos "cuasi ordenados", la ordenación debiera ser más rápida, pero el número de pasadas es fijo, completamente independiente del grado de orden o anarquía en que se encuentra la lista inicial.

Si tras una pasada ninguno de los elementos de la serie se ha movido, quiere decirse que la lista ya está ordenada, por lo que lo lógico es salir de los bucles, abandonando el proceso.

Este método incorpora una marca, consistente en hacer bascular el valor de una variable entre 0 y 1. Si hay cambio, toma el valor 0, si no lo hay toma el 1 (por supuesto que pueden tomarse dichos valores al revés, o de cualquier otra forma similar). El control de la marca puede realizarse mediante IF o bien aprovechando la potencia de la sentencia WHILE, como en el listado que se ofrece a continuación.

```
1000 ' ORDENACION. BURBUJA CON MARCA
1015 ' ........... ORDENA3 ..............
1020 MARCA=0
1030 WHILE MARCA=0
1040 MARCA=1
1050 FOR I=1 TO N-1
1060 IF A$(I)>A$(I+1) THEN X$=A$(I):A$(I
)=A$(I+1):A$(I+1)=X$:MARCA=0
1070 NEXT I
1080 WEND
```

Este método, habida cuenta de sus características, está especialmente indicado en listas con datos casi ordenados, como por ejemplo listas ordenadas, en las que se intercalan o sustituyen nuevos datos y deben volver a ordenarse (ficheros dinámicos).

5. Intercambio retardado

Una mejora de todos los métodos basados en la búsqueda del mínimo por intercambios sucesivos, consiste en retardar dichos intercambios, en tanto en cuanto que dicho mínimo no lo sea de toda la lista **(mínimo absoluto).**

Es decir, en cada pasada se extrae el mínimo absoluto. Una vez encontrado se sitúa en su sitio. De esta forma en la comparación de dos elementos no se produce intercambio, hasta haber encontrado dicho mínimo (retardo en el intercambio).

El proceso se repite nuevamente para encontrar el segundo mínimo, y así sucesivamente, la siguiente rutina emplea dicho método.

```
1000 ´ ORDENACION POR INTERCAMBIO RETARD
ADO
1015 ´ ........... ORDENA4 ..............
1020 FOR I=1 TO N-1
1030 MINIMO=I
1040 FOR J=I+1 TO N
1050 ´ Busqueda del minimo .....
1060 IF A$(J)<A$(MINIMO) THEN MINIMO=J
1070 NEXT J
1080 ´ Intercambio .............
1090 IF MINIMO-I THEN  THEN X$=A$(MINIMO
):A$(MINIMO)=A$(I):A$(I)=X$
1100 NEXT I
```

El bucle interno (líneas 1040-1070) es el encargado de encontrar el mínimo en cada pasada. Tras salir de este bucle, se produce la colocación del mínimo encontrado en el lugar que le corresponde. Observe que en la búsqueda del mínimo, lo que se recoge no es su valor, sino simplemente el lugar que ocupa.

6. Búsqueda del mínimo y máximo. Doble pasada

Si en el método anterior se buscaba el mínimo, parece lógico buscar simultáneamente el máximo. El mínimo irá al primer lugar y el máximo al último, por lo que la siguiente pasada debe realizarse sobre N-2 elementos, en vez de sobre N-1. En pasadas sucesivas, los límites del bucle serán cada vez menores, contando de dos en dos, todo lo cual da más velocidad al proceso.

Compare ambas rutinas y haga una prueba de velocidad, con un número de datos suficientemente grande.

```
1000 ´ ORDENACION METODO DOBLE PASADA
1015 ´ ........... ORDENA5 ...........
1020 PRIMERO=1:ULTIMO=N-1
1030 WHILE ULTIMO>PRIMERO
1040 FOR K=1 TO N-1
1050 IF A$(K)>A$(K+1) THEN X$=A$(K):
A$(K)=A$(K+1):A$(K+1)=X$:Y=K
1060 NEXT K
1070 ULTIMO=Y-1
1080 FOR K=ULTIMO TO PRIMERO STEP -1
```

```
1090 IF A$(K)>A$(K+1) THEN X$=A$(K):
A$(K)=A$(K+1):A$(K+1)=X$:Y=K
1100 NEXT K
1110 PRIMERO=Y+1
1120 WEND
```

7. Método de SHELL-METZNER

Todos los métodos expertos tienen el inconveniente común de su lentitud (en mayor o menor grado) cuando el número de datos es grande y se encuentran muy desordenados, debido a que el número de vueltas de los bucles resulta excesivamente elevado.

El **Shell-sort** -o método de "concha"- está basado en un algoritmo matemático, algo más complicado que la simple comparación e intercambio, ya que en él la comparación no se realiza como parece más lógico -entre elementos consecutivos- sino entre elementos separados entre sí en una distancia constante, que llamaremos CTE, cuyo valor más habitual es:

$$CTE = (2\uparrow X-1)/2$$

Donde X es el exponente de la menor de las potencias de 2, de forma que $2\uparrow X > N$ (número de elementos de la lista).

De esta forma, las comparaciones se harán entre

A$(K) y A$(K+CTE)

donde K representa la variable del bucle.

Aunque existen otros algoritmos para calcular CTE, el indicado es uno de los más utilizados, pues produce muy buenos resultados.

Veamos el listado:

```
1000 ' ORDENACION METODO SHELL (1)
1015 ' ........... ORDENA6 ........
1020 X=FIX (LOG(N)/LOG(2))
1030 CTE=(2↑X-1)/2
1040 WHILE CTE
1050 X=N-CTE:Y=1
1060 WHILE Z<=X
1070 K=Z
1080 IF A$(K)>A$(K+CTE) THEN X$=A$(K):
A$(K)=A$(K+CTE):A$(K+CTE)=x$:K=K-CTE:
IF K>1 THEN 1100
```

```
1090 Z=Z+1
1100 WEND
1110 CTE=CTE/2
1120 WEND
```

Una variante también muy utilizada es hacer que CTE valga

CTE = N/2

de forma que las comparaciones se realizarán

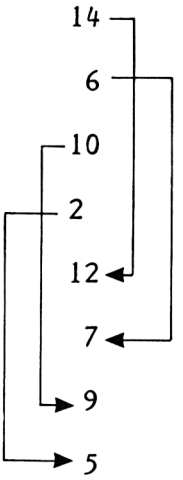
A$(K) y A$(K+N/2)

El funcionamiento es el siguiente: primero se comparan los elementos dos a dos en la forma indicada y se ordenan. Segundo, se fusionan los grupos de dos elementos en grupos de cuatro. Tercero, los grupos de cuatro en grupos de ocho y así hasta obtener la lista clasificada.

Como ejemplo gráfico, a fin de facilitar el seguimiento del método, veamos cómo se ordenaría la siguiente lista numérica:

14, 6, 10, 2, 12, 7, 9, 5

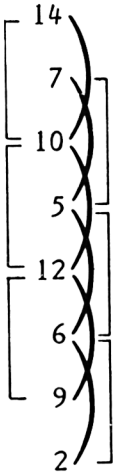
- valor de N=8
- valor de CTE=8/2=4



2-5
9-10
6-7
12-14

Grupo de 2

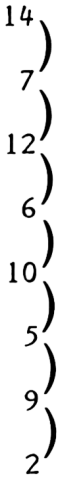
Comparación 1ª
CTE=4



2 - 5 - 6 - 7
9 - 10 - 12 - 14

Grupo de 4

Comparación 2ª



2 - 5 - 6 - 7 - 9 - 10 - 12 - 14

Grupo de 8 (ordenado)

Comparación 3ª

A continuación exponemos el listado de esta variante del método de Shell.

```
1000 ´ ORDENACION. METODO DE SHELL (2)
1015 ´ ........... ORDENA7 ...........
1020 CTE=INT((N+1)/2)
1030 FOR K=1 TO N-CTE
1040 IF A$(K)<=A$(K+CTE) THEN 1060
1050 X$=A$(K):A$(K)=A$(K+CTE):A$(K+CTE)=X$
1060 NEXT K
1070 IF CTE=1 THEN 1100 ELSE CTE=CTE-1:GOTO 1030
1100 ´ SALIDA DE DATOS
```

8. QUICKSORT

El método de ordenación Quicksort, último de los expuestos, es sin duda el más eficaz en el tratamiento de grandes cantidades de datos y muy desordenados.

La metodología no es difícil, se trata de realizar unas particiones en la lista, tratándolas por separado y con sucesivas subdivisiones. Se toma el primer elemento de la lista desordenada como pivote, se comparan con el resto de los elementos, colocándolos en dos grupos, los menores encima y los mayores debajo.

MENORES	ELEMENTO PRIMERO (PIVOTE)	MAYORES

A continuación se realiza la partición de la lista en dos y se vuelve a elegir un pivote por grupo, y por comparación se vuelve a obtener dos subconjuntos de cada uno de ellos y se repite la fragmentación y el proceso hasta que no pueden volverse a dividir.

Como crítica al método, debemos indicar que frente a una mayor optimización en el tratamiento de listas grandes y desordenadas, se encuentra el inconveniente de su mayor consumo de memoria al utilizar matrices auxiliares, aunque hoy día las memorias de los ordenadores han dejado de ser problema.

Veamos el listado de esta rutina.

```
1000 ' METODO DE ORDENACION QUICKSORT
1015 ' ........... ORDENA8 ...........
1020 CON=FIX(LOG(N)/LOG(2))+4
1030 ERASE X:DIM X(CON,2)
1040  X(1,1)=1:X(1,2)=N
1050 WHILE CON
1070 C1=X(CON,1):F1=X(CON,2)
1080 CON=CON-1
1090 X$=A$(C1):C2=C1
1100 F2=F1:F1=F1+1
1110 F1=F1-1
1120 IF F1=C1 THEN 1170
1130 IF A$(F1)<X$ THEN A$(C1)=A$(F1) ELSE 1110
1140 C1=C1+1
1150 IF C1=F1 THEN 1170
```

```
1155 IF A$(C1)>X$ THEN A$(F1)=A$(C1):GOTO 1110 ELS
E 1140
1160 IF A$(C1)>X$ THEN A$(F1)=A$(C1):GOTO 1110
1170 A$(C1)=X$
1180 IF C2<C1-1 THEN X(CON+1,1)=C2:X(CON+1,2)=C1-1
:CON=CON+1
1190 IF F1+1<F2 THEN X(CON+1,1)=F1+1:X(CON+1,2)=F2
:CON=CON+1
1200 WEND
```

9. Apéndice: Programa general de Ordenación

Como ya se indicó en el epígrafe 1 de este capítulo, y con el fin de que el lector pueda probar (si lo desea) los distintos métodos de ordenación expuestos, se incluyen en este apéndice una colección de rutinas que permiten trabajar en este campo de la ordenación y con cualquiera de los métodos.

Las rutinas se han preparado de forma que esté asegurada su compatibilidad conjunta. Como referencia incluimos a continuación su número de orden, nombre y números de línea que abarca cada una de ellas.

Rutina	Nombre	Nº Línea	Comentarios
1ª	MENU	10 - 290	Menú de opciones que controla funcionamiento conjunto
2ª	ENTRADAT	300 - 430	Entrada de datos (teclado)
3ª	GRAFICHE	500 - 620	Salva el fichero en disco/cinta
4ª	LEEFICHE	700 - 820	Lee el fichero de disco/cinta
5ª	ORDENAn	1000 - ——	Rutina de ordenación nº n
6ª	SALDATOS	2000 - 2070	Salida de datos ordenados

A continuación se incluyen las citadas rutinas listadas y comentadas.

1ª MENU. Consiste en un menú directorio desde el cual puede accederse a todas las demás rutinas. Contiene las siguientes opciones:

1. ENTRADA DE DATOS
2. GRABAR FICHERO
3. LEER FICHERO
4. ORDENACION
5. SALIDA DATOS ORDENADOS

6. CAT (Catálogo)
7. FIN

Las opciones 1 a 4 se corresponden obviamente con las rutinas del cuadro anterior (2ª a 5ª).

La opción 6 se incluye con el fin de facilitar el directorio del disco sin tener que detener el programa. De esta forma podemos ver la evolución de los ficheros en los que estamos trabajando. El listado se incluye en el ensamblado final.

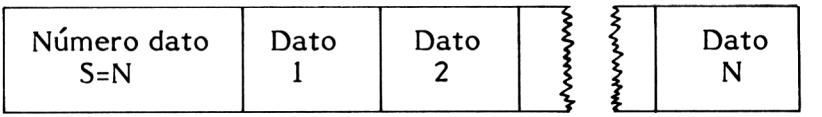
2ª ENTRADA DE DATOS (ENTRADAT). Permite la introducción de datos por teclado -de una lista de la dimensión deseada- tanto numéricos como alfanuméricos. Este tipo de entradas fue estudiado en el capítulo 2º y no incorpora depuración de los mismos, por lo que deben ser introducidos con cierto cuidado (puede añadirse una rutina de depuración de las estudiadas en el capítulo 2).

```
  300 ´ RUTINA DE ENTRADA DE DATOS
310 ´ ........ ENTRADAT ................
....
320 MODE 1:WINDOW R1,1,40,25,25
330 INPUT "Numero de datos de la lista "
;N
340 DIM A$(N)
350 PRINT "INTRODUZCA LOS DATOS"
360 FOR F=1 TO N
370 PRINT F;". ";:INPUT D$
380 IF VAL(D$) THEN IF LEN(D$)>LON THEN
LON=LEN(D$)
390 A$(F)=D$
400 NEXT F
410 FOR F=1 TO N:IF VAL(A$(F))=0 AND A$(
F)<>"0" THEN 430
420 A$(F)=STRING$(LON-LEN(A$(F)),"0")+A$
(F)
430 NEXT F
```

3ª GRABAR EN FICHERO (GRAFICHE). Los datos introducidos mediante **ENTRADAT** son grabados en el disco/cinta, de forma secuencial.

En primer lugar se guarda el número de datos que contiene la lista (N) y después los datos propiamente dichos.

De esta forma, el valor de N también se incluye en el fichero.



El nombre del fichero se establece mediante INPUT con el fin de que la entrada de datos sea válida para números y palabras. En las líneas 110 a 130 se uniformizan las cadenas para que en su posterior ordenación no se produzcan errores (datos numéricos).

```
500 'GRABACION DE DATOS EN FICHERO
505 '.......... GRAFICHE ..............
520 CLS:PRINT "GRABACION DE LOS DATOS EN
 FICHERO"
530 PRINT "==============================
====="
540 INPUT "NOMBRE DEL FICHERO (max 8)";F
$
550 IF LEN (F$)>8  THEN PRINT "Nombre de
masiado largo":GOTO 540
555 PRINT "G R A B A N D O"
560 OPENOUT F$
570 WRITE #9,N
580 FOR K=1 TO N
590 WRITE #9,A$(K)
600 NEXT K
610 CLOSEOUT
620 PRINT CHR$(24);"FICHERO  ";F$;"  GRA
BADO";CHR$(24)
```

4ª LEER DATOS DE UN FICHERO (LEEFICHE). Permite realizar la operación contraria a la rutina anterior, es decir, recuperar los datos guardados en el disco/cinta.

Para poder realizar la lectura se pide el nombre del fichero, que debe ser el mismo que se utilizó al grabar.

Como el primer dato del archivo es el número de elementos, los bucles y dimensionamiento de la matriz (lista) que va a recibir los datos se realiza en función de dicho número, leído del fichero por la variable (N).

```
700 ´LECTURA DE DATOS DE UN FICHERO
710 ´......... LEEFICHE ..............
720 INPUT "NOMBRE DEL FICHERO (max 8)";F
$
730 IF LEN (F$)>8  THEN PRINT "Nombre de
masiado largo":GOTO 720
740 PRINT "L E Y E  N D O"
750 OPENIN F$
760 INPUT R9,N:DIM A$(N)
770 FOR K=1 TO N
780 INPUT R9,A$(K)
790 NEXT K
800 CLOSEIN
810 PRINT CHR$(24);"FICHERO ";F$;"LEIDO"
;CHR$(24)
820 PRINT :PRINT "PRESIONE UNA TECLA PAR
A CONTINUAR":CALL &BB18
```

5ª ORDENACION (ORDENAn). Esta rutina puede ser cualquiera de las incluidas en los apartados precedentes.

Los nombres dados a dichas rutinas es ORDENA, seguido de un número que las identifica y cuya relación incluimos a continuación.

Nombre	Método
ORDENA1	Intercambio
ORDENA2	BUBBLE
ORDENA3	BUBBLE con marca
ORDENA4	Intercambio retardado
ORDENA5	Doble pasada
ORDENA6	SHELL-METZNER (1)
ORDENA7	SHELL-METZNER (2)
ORDENA8	QUICK-SORT

6ª SALIDA DE DATOS ORDENADOS (SALDATOS). Constituye una sencilla rutina de salida de datos ordenados. En caso de que el número de datos sea mayor de 20, se interrumpe el scrolling automático de pantalla, a fin de observarlos con detenimiento, y presionando una tecla se produce la salida del siguiente bloque de 20 datos.

```
2000 ' RUTINA DE SALIDA DE DATOS ORDENAD
OS
2005 ' ......... SALDATOS ............
..
2010 CLS
2020 PRINT "    D A T O S    O R D E N A
 D O S"
2030 PRINT "============================
==========="
2040 FOR F=1 TO N
2050 IF F/20=INT (F/20) THEN PRINT R1,"P
RESIONE TECLA PARA CONTINUAR LISTADO":CA
LL &BB18:CLS
2060 PRINT F;". ";A$(F)
2070 NEXT F
```

Conjunto Ensamblado

Por último y para mayor comodidad del lector, se incluye un programa con todas las rutinas mencionadas, convenientemente ensambladas y regidas desde el menú.

Como observará, se han añadido algunas pequeñas modificaciones con el fin de que puedan funcionar en conjunto.

Para probar cómodamente todos los métodos de ordenación descritos, e incluso cronometrar el tiempo invertido por cada uno, proceda de la siguiente forma.

1. Grabe las rutinas de ordenación con el nombre que se incluye en la línea 1015 de cada listado.

2. Grabe el programa que incluimos al final con las rutinas de trabajo ensambladas, tal y como está en el listado. El nombre indicado para este programa es **ORDEN.**

3. Haga RUN"ORDEN o RUN simplemente si lo tiene ya en memoria. El programa le pedirá el nombre de la rutina de ordenación que desea utilizar; introduzca el nombre según figura en el epígrafe 5º (si los grabó con esos mismos). Se producirá un MERGE para efectuar la fusión del programa con la rutina de ordenación seleccionada. Dicho MERGE lleva un parámetro de puesta en marcha, por lo que tras la carga, el programa continúa su curso.

Seleccione tras esto la opción 1 e introduzca una lista de datos numéricos (del tamaño que desee) totalmente desordenados. (También puede usar un método aleatorio sustituyendo el INPUT de **ENTRADAT** por un RND).

4. Cuando retorne al MENU, seleccione la opción 2 para grabarlos en el disco/cinta. Como nombre puede utilizar "NUMEROS" o cualquier otro.

5. Seleccione nuevamente la opción 1 e introduzca ahora una lista de datos alfanuméricos. Después la 2 para grabarlos. Un nombre apropiado puede ser, por ejemplo, "PALABRAS".

6. Ahora puede proceder a ordenar cualquiera de los dos ficheros creados. Si es precisamente el último, ésta se encuentra ya en memoria; por tanto, basta con que seleccione la opción 4 y después la 5 con el fin de verlos ordenados.

7. Si desea ordenar un archivo que está contenido en disco o cinta, utilice la opción 3 para leerlos, luego 4 para ordenar y 5 para imprimirlos en pantalla, ya ordenados.

Tenga en cuenta que todas las rutinas retornan al MENU al finalizar su tarea.

Notas:

1 En todos los métodos de ordenación expuestos, se ha considerado que ésta se realiza de menor a mayor. Para obtener clasificaciones en sentido contrario basta con cambiar los operadores de relación de los IF, o bien, simplemente cambie el bucle de salida de datos en la forma.

```
FOR K=N TO 1 STEP-1
```

(Línea 2040 de la rutina SALDATOS).

2 Si desea salvar los datos ordenados tras las operaciones pertinentes, utilice la opción 2 (grabar), ya que tras la opción 4 (ordenar), en memoria estarán todos los datos ordenados.

```
10 ' CONJUNTO DE RUTINAS PARA LA UTILIZACION DE LOS PROGRAMAS DE ORDENACION
15 ' ............ ORDEN ...............
20 MODE 1:DIM A$(100):DIM X(2,1)
30 PRINT "   INTRODUZCA EL NOMBRE DEL PROGRAMA DE ORDENACION DESEADO."
35 INPUT O$
40 PRINT :PRINT "   ESTE SERA ENSAMBLADO CON LAS PRESEN -TES RUTINAS PARA SU UTI
LIZACION CONJUNTA"
50 PRINT "ARRANQUE DENUEVO CON";CHR$(24);" GOTO 80 ";CHR$(24)
60 DELETE 1000-1700
80 CHAIN MERGE O$+".BAS",100
100 '        M   E   N   U
110 ' Preparado para trabajar con las rutinas ENTRADAT,GRAFICHE,LEEFICHE,ORDENAn
, Y SALDAT
130 MODE 1:RESTORE
135 READ X$:LOCATE 12,1:PRINT X$
136 LOCATE 11,2:PRINT "============"
140 FOR K=4 TO 16 STEP 2
150 READ X$
160 LOCATE 6,K:PRINT K/2-1;". ";X$
170 NEXT K
180 DATA M  E  N  U,ENTRADA DATOS,GRABAR EN FICHERO,LEER UN FICHERO,ORDENACION,S
ALIDA DATOS ORDENADOS,CAT (catalogo),F  I  N
190 ' Seleccion
200 WINDOW #1,1,40,25,25
210 INPUT #1,"SELECCIONAR OPCION";OP
220 IF OP<1 OR OP>7 THEN PRINT CHR$(7):GOTO 210 ELSE CLS
230 ON OP GOSUB 300,500,700,1000,2000,280,290
240 GOTO 100
250 CLS #1:PRINT #1,"PRESIONE UNA TECLA PARA CONTINUAR":CALL &BB18:RETURN
280 CAT:GOSUB 250:RETURN
290 END
300 ' RUTINA DE ENTRADA DE DATOS
310 ' ........ ENTRADAT ....................
320 MODE 1:WINDOW #1,1,40,25,25
330 INPUT "Numero de datos de la lista ";N
340 ERASE A$:DIM A$(N)
350 PRINT "INTRODUZCA LOS DATOS"
360 FOR F=1 TO N
370 PRINT F;". ";:INPUT D$
380 IF VAL(D$) THEN IF LEN(D$)>LON THEN LON=LEN(D$)
390 A$(F)=D$
400 NEXT F
410 FOR F=1 TO N:IF VAL(A$(F))=0 AND A$(F)<>"0" THEN 430
420 A$(F)=STRING$(LON-LEN(A$(F)),"0")+A$(F)
430 NEXT F
435 FOR K=1 TO N:PRINT A$(K),:NEXT
440 PRINT "ENTRADA DE DATOS TERMINADA"
450 GOSUB 250
460 RETURN
```

```
500 'GRABACION DE DATOS EN FICHERO
505 '........... GRAFICHE ...............
520 CLS:PRINT "GRABACION DE LOS DATOS EN FICHERO"
530 PRINT "================================"
540 INPUT "NOMBRE DEL FICHERO (max 8)";F$
550 IF LEN (F$)>8  THEN PRINT "Nombre demasiado largo":GOTO 540
555 PRINT "G R A B A N D O"
560 OPENOUT F$
570 WRITE #9,N
580 FOR K=1 TO N
590 WRITE #9,A$(K):PRINT A$(K)
600 NEXT K
610 CLOSEOUT
620 PRINT CHR$(24);"FICHERO  ";F$;"  GRABADO";CHR$(24)
630 GOSUB 250
640 RETURN
700 'LECTURA DE DATOS DE UN FICHERO
710 '......... LEEFICHE ..............
720 INPUT "NOMBRE DEL FICHERO (max 8)";F$
730 IF LEN (F$)>8  THEN PRINT "Nombre demasiado largo":GOTO 720
740 PRINT "L E Y E  N D O"
750 OPENIN F$
760 INPUT #9,N:PRINT "NUMERO DE DATOS=";N
765 ERASE A$:DIM A$(N)
770 FOR K=1 TO N
780 INPUT #9,A$(K)
790 NEXT K
795 FOR K=1 TO N:PRINT A$(K):NEXT
800 CLOSEIN
810 PRINT CHR$(24);"FICHERO ";F$;" LEIDO";CHR$(24)
820 GOSUB 250
830 RETURN
1000 ' RUTINA DE ORDENACION
1800 PRINT "D A T O S   O R D E N A D O S"
1810 GOSUB 250
1900 RETURN
2000 ' RUTINA DE SALIDA DE DATOS ORDENADOS
2005 ' ........... SALDATOS ...............
2010 CLS
2020 PRINT "    D A T O S    O R D E N A D O S"
2030 PRINT "======================================"
2040 FOR F=1 TO N
2050 IF F/20=INT (F/20) THEN PRINT #1,"PRESIONE TECLA PARA CONTINUAR LISTADO":CA
LL &BB18:CLS
2060 PRINT F;". ";A$(F)
2070 NEXT F
2080 GOSUB 250
2090 RETURN
```

CAPITULO 4

FICHEROS SECUENCIALES

1. **Introducción**

2. **Conceptos previos**

   * Apertura y cierre de ficheros
   * Lectura de datos
   * Almacenamientos de datos

3. **Ficheros en cinta**

   * Conceptos básicos
   * Estudio del almacenamiento de datos en cassette
   * El azimut de la cabeza lectora-grabadora

4. **Ficheros en disco**

   * Generalidades
   * Borrar ficheros del disco
   * Listar el contenido de un fichero (CP/M)

5. **Apéndice. Programa Agenda (para cinta o disco)**

Ficheros Secuenciales

1. Introducción

Como ya se indicó en el capítulo 1, este tipo de ficheros consiste en el almacenamiento de los datos, uno a continuación de otro, es decir, en **forma secuencial.** También se describió que su principal inconveniente consiste en que para acceder a un dato en concreto, deben leerse todos los almacenados o, al menos, todos los anteriores al buscado.

A pesar de este inconveniente, son útiles en muchas aplicaciones y por otro lado, es el único tipo de fichero que permiten los Amstrad desde el BASIC.

Existe una aplicación a la que dedicamos el capítulo siguiente que permite trabajar con ficheros de **acceso directo,** pero no viene implantado con el lenguaje, sino que se carga desde disco, obteniéndose unas nuevas órdenes extendidas.

Hay que tener en cuenta que los únicos ficheros posibles en cinta son los secuenciales, debido a las propias características del cassette incorporado al Amstrad CPC464.

2. Conceptos previos

Apertura y cierre de ficheros

Existen dos conceptos básicos en el manejo de ficheros secuenciales:

1º Abrir el fichero

2º Cerrar el fichero

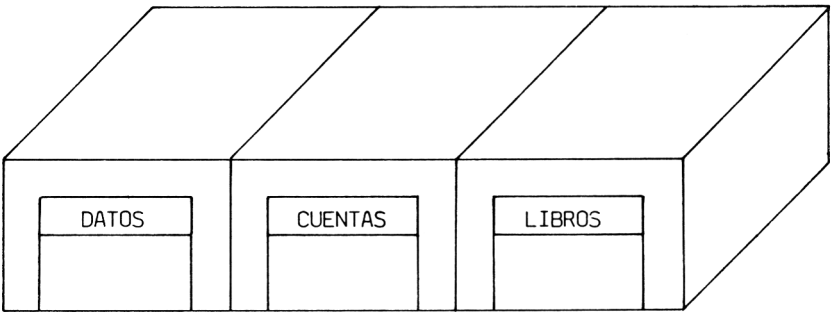
Ambas operaciones se subdividen en dos clases según el trabajo que se vaya a realizar con los mismos, y reciben el siguiente nombre en BASIC.

OPERACION		DESTINO	COMANDO
Abrir el fichero	hacia afuera	(disco/cinta)	OPENOUT
	hacia dentro	(ordenador)	OPENIN
Cerrar el fichero	hacia afuera	(disco/cinta)	CLOSEOUT
	hacia dentro	(ordenador)	CLOSEIN

Veamos los significados de los conceptos y comandos indicados en el cuadro anterior.

* Abrir un fichero

Un fichero es una estructura cerrada dotada de una "puerta" y un final -o "fondo de saco"- sin salida. Dicha puerta, además tiene un nombre clave que permite su apertura.



Para poder acceder a los datos contenidos en un fichero, primero hay que abrir dicha puerta mediante su clave; después podemos sacar o introducir datos en este almacén.

Ahora bien, puesto que se pueden realizar dos operaciones -sacar datos o introducirlos- debe indicarse qué tipo de operación va a realizarse, ello da lugar a dos comandos:

OPENOUT Abrir el fichero para grabar en él datos.

OPENIN Abrir el fichero para leer datos contenidos en él.

Estos dos comandos de apertura deben llevar a continuación el nombre que identifica al fichero (para que pueda abrirse).

Si se trata de OPENOUT y no existía previamente el fichero, el ordenador, a la vez que realiza la operación de apertura, lo crea.

El nombre del fichero puede indicarse entre comillas, o bien con una variable alfanúmerica que lo contenga. Por ejemplo:

```
OPENOUT "nombre"
OPENOUT A$          siendo A$ = "nombre"
```

Las limitaciones del nombre de los ficheros son las mismas que para los nombres de los programas, 8 caracteres (sin espacios ni símbolos) y además puede indicarse el tipo (máx. 3 caracteres) en la forma:

```
PALABRAS.TIP
```

al igual que se usa en CP/M.

* Cerrar un fichero

Es la operación contraria a la apertura. Los dos comandos derivan de CLOSE (cerrar), la segunda parte de la palabra de los mismos, hace alusión al tipo de apertura previo utilizado; es decir, existen dos comandos de cierre: CLOSEOUT y CLOSEIN, que corresponden respectivamente.

Apertura	Cierre
OPENOUT	CLOSEOUT
OPENIN	CLOSEIN

A diferencia de las aperturas, las órdenes de cierre no necesitan indicar el nombre del fichero.

Cuando se produce el cierre de un fichero, el ordenador añade una señal de "fin de fichero", cuyo nombre en BASIC es EOF.

Esta señal puede ser verificada e interpretada, por ejemplo con un INPUT, teniendo en cuenta que EOF toma únicamente los valores -1 y 0.

Valor de EOF	Significado	Comentario
-1	verdadero	No hay ningún fichero abierto o se ha alcanzado el final del fichero
0	falso	Existe un fichero abierto o no se ha alcanzado su final.

Esta señal permite leer un fichero del que desconocemos el número de elementos que lo forman.

La rutina que indicamos a continuación, permite mostrar en pantalla el contenido de un fichero de forma similar a como lo hace la orden TYPE de CP/M.

```
10 ' MOSTRAR EL CONTENIDO DE UN FICHERO
20 ' ........... FS0 ....................
30 INPUT "NOMBRE DEL FICHERO";F$
40 OPENIN F$
50 WHILE NOT EOF
60 INPUT #9,A$:PRINT A$
70 WEND
80 CLOSEIN
```

* Cauces

Tanto el disco como la cinta están señalados en los Amstrad CPC como cauce número 9, indicado mediante #9. Como recordará, todos los cauces disponibles son:

Cauce	Destino
#0	Pantalla
#1	Ventana 1
#2	.
.	.
.	.
#7	Ventana 7
#8	Impresora
#9	Cinta/Disco

Los modelos CPC664 y 6128 pueden trabajar tanto en cinta (con un cassette supletorio), como con discos (el de origen e incluso adicionales). La forma de distinguir la cinta del disco es indicar previamente el dispositivo.

ORDEN	DISPOSITIVO
¦ TAPE	Cinta
¦ DISC	Disco

Cuando se encienden o inicializan estos modelos, el dispositivo por omisión es el disco.

Lectura de datos

* Lectura con FOR

Existen dos sentencias BASIC utilizadas para la lectura y escritura en relación con los ficheros, acompañados del cauce cinta/disco, que -como hemos visto anteriormente- es el nueve (#9).

Dichas sentencias, son las siguientes:

INPUT # 9	Leer del fichero
WRITE # 9	Escribir en el fichero

La sentencia WRITE #9, puede ser sustituida por PRINT #9, que produce el mismo efecto, aunque la apropiada es sin duda la primera.

La sintaxis de estas instrucciones es simple:

INPUT #9, variable o lista de variables

La variable indicada es la encargada de almacenar el dato leído del fichero, de la misma manera que en INPUT, la variable almacena el dato leído por el teclado.

La naturaleza de esta variable (numérica/alfanumérica) dependerá del tipo de datos a leer y almacenar, es decir, la misma que se utilizó en la grabación.

Tras un INPUT #9, pueden incluirse varias variables en la forma:

INPUT #9, var1, var2, var3,...varn

Como veremos más adelante en los ejemplos, las variables más utilizadas para leer datos de un fichero son las matriciales (indexadas) ya que éstas permiten, por un lado, almacenar gran cantidad de datos; y por otro, aprovechar un bucle como estructura de lectura, de tal forma que la variable del FOR sea utilizada como índice de la matriz.

Veamos un ejemplo de rutina de lectura, con variable indexada numérica:

```
10 ´ RUTINA DE LECTURA CON FOR
15 ------------FS1------------
20 N=30:DIM X(N)
30 OPENIN "DATOS"
40 FOR K=1 TO N
50 INPUT #9,X(K)
60 NEXT K
70 CLOSEIN
```

En el ejemplo, se supone que el fichero contiene 30 datos numéricos y que van a ser almacenados en la matriz X.

* Lectura con contador

Aunque la estructura FOR es cómoda, puede sustituirse también por un contador controlado por un IF, es decir, los programas que exponemos a continuación son equivalentes.

```
10 FOR K=1 TO 100
20 PRINT K
30 NEXT K
40 END
```

```
10 ´ ........ FS1.2 ............
20 K=1
30 PRINT K
40 K=K+1: ´Contador
50 IF K>100 THEN END
60 GOTO 20
```

Ambos programas, como puede comprobar, escriben las números del 1 al 100.

Si esta idea la unimos a lo indicado para EOF (fin de fichero), podemos leer los datos de un fichero sin necesidad de conocer cuántos datos son. Para ello, la dimensión de la matriz habrá de ser superior al número de datos, es decir, que aunque no conozcamos el número de ellos con exactitud, sí debemos conocer el orden de magnitud.

Así, si sabemos que un fichero contiene entre 70 y 90 datos, bastará con dimensionar 100 para que todos quepan en la matriz.

Una rutina de lectura con contador y control de EOF sería:

```
10 ' RUTINA DE LECTURA CON CONTADOR
20 '      CONTROL CON EOF
30 ' ........... FS2 ................
40 DIM A(100)
50 OPENIN "DATOS"
60 K=1
70 INPUT #9,A(K)
80 IF EOF THEN 100
90 K=K+1:GOTO 70
100 CLOSEIN
```

La línea 80 controla el fin de fichero. Si el IF es cierto, se cierra el fichero y finaliza el programa; en caso contrario se incrementa el contador (índice de la matriz) y continua la lectura.

* Lectura con WHILE

Aprovechando las características de los bucles WHILE; cuyo número de vueltas no es fijo como ocurre con FOR, sino que permanece el bucle mientras se de una determinada condición, podemos pues, elaborar una rutina de lectura, apoyada además en la señal EOF.

```
10 ' RUTINA DE LECTURA CON WHILE
20 ' ........... FS3 ................
30 DIM A(100)
40 OPENIN "DATOS"
50 WHILE NOT EOF
60 K=K+1
70 INPUT #9,A(K)
80 WEND
100 CLOSEIN
```

Cualquiera de los tres métodos expuestos es válido, aunque quizá el último sea el más elaborado.

Hemos comentado la necesidad de conocer el número de elementos del fichero de forma exacta, como en la rutina primera, o aproximada, como en el de las dos últimas.

La experiencia nos dice que, salvo programadores "muy ordenados", que guardan sus datos y programas con información abundante, lo normal es que al cabo de cierto tiempo, uno no sepa ni siquiera el orden de magnitud del fichero, lo cual obliga a dimensionamientos excesivos de la matriz, que a fin de cuentas son grandes "comedoras" de memoria.

La solución a este problema es bien sencilla. Acostumbrémonos a grabar como primer dato del fichero, el número de elementos que lo componen; de esta forma, la lectura del primer dato nos permitirá realizar un dimensionamiento óptimo y, en caso de utilizar un FOR, conocer su límite.

Una rutina, al efecto, puede ser:

```
10 ' RUTINA DE LECTURA CONOCIDO EL
20 ' NUMERO DE DATOS (PRIMER ELE.)
30 ' ........... FS4 ..............
40 OPENIN "DATOS"
50 INPUT #9,N
60 DIM A(N)
70 FOR K=1 TO N
80 INPUT #9,A(K)
90 NEXT K
100 CLOSEIN
```

Almacenamiento de datos

Como se indicó en el apartado anterior, la sentencia BASIC de escritura en archivo (grabación) es WRITE #9.

La sintaxis es igual que la de lectura

WRITE #9, variable

o bien con varias variables separadas por comas

WRITE #9, var1, var2, var3,...,varn

Como el número de datos a almacenar es conocido generalmente, lo más sencillo resulta utilizar una rutina de grabación utilizando un bucle FOR.

Por ejemplo:

```
10 ' RUTINA DE GRABACION CON FOR
20 ' ........... FS5 ............
30 OPENOUT "DATOS"
40 FOR K=1 TO N
50 WRITE #9,A(K)
60 NEXT K
70 CLOSEOUT
```

Siendo **n** el número de datos a salvar y A() la matriz de datos.

Ahora bien, en muchas ocasiones, por ejemplo al confeccionar una agenda, se prepara una matriz grande en la que se van almacenando los datos de una ficha, pero resulta que el número de dichos datos es inferior a la dimensión original. Si tomamos dicha dimensión como límite del FOR resultará que almacenaremos un montón de datos vacíos.

Esto puede evitarse mediante un IF, de tal forma qe si se encuentra una ficha sin datos, se abandone la grabación.

Veamos todo esto con un ejemplo completo.

Supongamos que deseamos almacenar en un fichero el nombre de los alumnos de una clase, que como máximo puede contener 50 alumnos, pero que en el momento en que se prepara el fichero sólo se han matriculado 8.

Preparemos dos rutinas una de entrada de datos y otra de almacenamiento.

```
10 ' CARGA Y GRABACION DE MATRICES
20 '     CON DATOS INCOMPLETOS
30 ' ........... FS6 ..............
40 AL=50:DIM AL$(AL)
50 FOR K=1 TO AL
60 PRINT "ALUMNO ";K;
70 INPUT AL$(K)
80 IF AL$(K)="" THEN 100
90 NEXT K
100 OPENOUT "ALUMNOS"
```

```
110 FOR K=1 TO AL
120 IF AL$(K)="" THEN 150
130 WRITE #9,AL$(K)
140 NEXT K
150 CLOSEOUT
```

Como puede observarse, tanto en la entrada de datos como en la rutina de grabación (100-150) se ha utilizado un control, de forma que si el contenido es "nada", se interrumpe el proceso. La introducción de "nada" en la línea 70, es equivalente a presionar ENTER, tras la aparición del INPUT.

En la línea 110, si va unida a las anteriores como programa "completo", podía haberse cambiado el límite final del FOR puesto que dicho límite (8 en el ejemplo) es conocido. Tenga en cuenta que de la línea 80 se sale con un valor superior en una unidad al número de alumnos que se han introducido. Por tanto, cabe modificar la segunda rutina, en la forma siguiente:

```
110 FOR J=1 to K-1
```

y eliminando el IF de la 120, ahora innecesario, puesto que el número de vueltas del bucle J será exactamente 8 (K-1=8).

A pesar de lo dicho, el método resulta muy eficaz en programas más complejos, en los que continuas modificaciones del fichero hagan difícil llevar la cuenta del número de datos.

En el caso de datos numéricos, la condición "nada" ("""), debe cambiarse por cero (0), siempre y cuando este número no pertenezca a la lista como valor posible. En caso contrario debe utilizarse en valor singular, por ejemplo -1, -9999, etc., sobre el que tengamos la certeza de su no existencia.

La estructura WHILE también es válida para la grabación de datos. En el caso comentado anteriormente de matrices grandes con datos incompletos, puede establecerse el bucle en la siguiente forma.

```
WHILE A$(K) <> ""        (cadenas)
WHILE A(K) > 0           (números)
```

Aunque en todo momento y por razones de claridad, nos hemos referido en nuestros ejemplos a listas y no a tablas. En los apéndices se incluyen programas completos de manejo de ficheros que utilizan tablas de dos dimensiones.

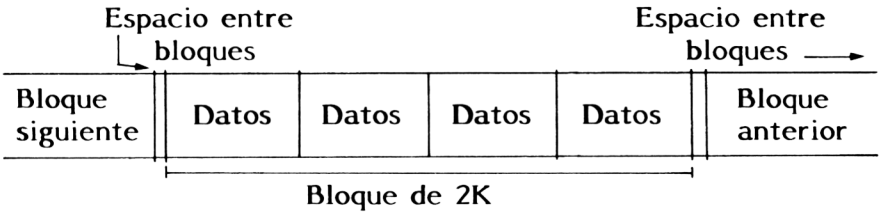
3. Ficheros en cinta

Conceptos básicos

El manejo de ficheros en cinta tiene sus peculiaridades, derivadas de las características de la unidad incorporada por el 464 y de las de un cassette convencional en los otros modelos.

Como sólo existe una unidad, estamos obligados a cambiar frecuentemente de cinta y a rebobinados y avances continuos. Para ello debe utilizarse el contador de vueltas, de forma que todo programa o fichero debe relacionarse en una lista con indicación del nombre exacto con que se grabó y los números de vuelta de comienzo y fin de la grabación, partiendo del 000 con la cinta totalmente rebobinada, única forma de no realizar un almacenamiento caótico, que dé lugar a pérdidas elevadas de tiempo y en muchos casos a la pérdida de los propios programas y/o ficheros.

Los datos son almacenados y leídos en el cassette en bloques de 2 K, que son fácilmente manejables y ocupan poca memoria. La forma en que se graban en la cinta se indica gráficamente en la figura siguiente.



Los espacios entre bloques permiten el arranque y parada del cassette, cada 2 K.

La forma en que se realiza la grabación por bloques, es a través de una memoria intermedia. Esta recibe los datos uno a uno y cuando se llena, se produce la grabación. También se produce la grabación de un bloque incompleto, cuando recibe la señal de cierre del fichero.

Para leer utiliza la misma técnica, lo cual evita el arranque y parada continua, si lo realizarse dato a dato.

Si utiliza un 664 o 6128 y quiere añadir una cassette externa, asegúrese que ésta dispone de una conexión de control remoto, para arranque y parada, porque en caso contrario, tendrá que controlar todo el proceso a mano.

La lentitud de trabajo de los ficheros sencuenciales en cinta, hace que el procesado dinámico sea prácticamente imposible, y lo más lógico es cargar todo el fichero en memoria, al comenzar la jornada de trabajo, efectuar todos los manipulaciones en sus datos y proceder a su grabación al final de la jornada, evitando las grabaciones intermedias. Desde luego si falta la alimentación, se habrá perdido todo el trabajo realizado hasta entonces con los datos.

La velocidad de grabación en cassette puede elegirse entre dos opciones, manejadas por la instrucción SPEED WRITE, los valores son 0 o 1, que tienen el siguiente significado.

SPEED WRITE 0	Velocidad lenta pero de máxima seguridad
SPEED WRITE 1	Velocidad rápida pero de menor seguridad.

En el apéndice 1 de este apartado incluimos un programa completo de manejo de archivos en cinta, válido tanto para el CPC464, como para los 664 y 6128 con cassette adicional o disco.

Se trata de una agenda de direcciones y teléfonos. Para una utilización más cómoda, introduzca todos los datos en mayúsculas.

Estudio del almacenamiento de datos en cassette

Este apartado está dedicado a aquellos lectores que quieran profundizar en el sistema de almacenamiento de datos en cassette, de la forma en que es realizada por el Amstrad, tanto en el CPC464 con cassette incorporado, como en los CPC664 y 6128 con cassette convencional, externo.

El cassette o radiocassette convencional, no nació precisamente pensado para aplicaciones informáticas, ni como periférico de un ordenador. La cinta cassette fue diseñada para recoger información de tipo analógico, voces y música, así como para reproducirla, a través de un altavoz, o bien enviarla, vía audio, a otros dispositivos de la misma naturaleza: otros cassettes, altavoces externos, amplificadores, etc.

Pero la aparición de los primeros ordenadores domésticos, donde el precio imponía una limitación en los periféricos, llevó a los diseñadores a pensar en la posibilidad de utilizar estos cassettes como medio de almacenamiento de información masivo y de bajo coste.

El problema que se planteó, fue que mientras la música se graba de forma analógica, el ordenador utiliza una información de tipo digital. (Tenga en cuenta que en sus más profundas raíces el ordenador sólo entiende y manipula ceros y unos).

Para la resolución del problema, se consideró que podían establecerse dos estados de sonido, lo más opuestos entre sí. A uno se le asignaría el valor 1 y al otro el 0 (así como grave y agudo o alto y bajo).

El Amstrad también se basa en este punto de partida. Detecta las inversiones de tono que esconden tras de sí un **bit** (unidad elemental de información). Si la duración de la inversión la denominamos DI, el ordenador la interpretará como un 0; por el contrario si es el doble, es decir, 2 x DI, la interpreta como un 1.

Como una cassette dispone de una sola cabeza de lectura/grabación, sólo puede grabar o leer un dato, es decir, un bit y no una secuencia de ellos como sería un **byte** (8 bits). De manera que toda la información se lee y almacena de forma secuencial, bit a bit y habida cuenta del carácter físico lineal de la cinta, para acceder en un determinado bit o área de bits, es necesario leer antes toda la anterior información, o bien recurrir a avance o rebobinado de la cinta, ayudado por el contador, a fin de situar la cabeza lo más próxima del área deseada.

El procedimiento en sí no es dificultoso, pero si lo es el hecho del gran número de errores que se pueden producir debido a gran cantidad de factores, como son: calidad de la cinta, velocidad de lectura/grabación, ruidos perturbadores, cintas poco apropiadas al tratamiento de datos digitales, etc.; y un último debido al azimut de la cabeza, que explicamos con detalle al final del apartado.

El CPC464, que incorpora una unidad de cassette, especialmente diseñada para dicho ordenador, tiene un margen de error muy pequeño y, sin duda, sorprenderá la gran fiabilidad del mismo a los usuarios que anteriormente hayan trabajado con otros ordenadores apoyados en cassettes convencionales.

En dicho modelo puede decirse que no se producen errores de lectura, cuando el programa o fichero ha sido grabado por el mismo, u otro de la serie, supuesto el buen trato y calidad de la cinta.

Otra historia, por otra parte muy de moda, es la gran cantidad de errores que se detectan, cuando se trata de leer el típico programa de juegos, de tipo comercial y que no es original, es decir, "pirata".

No vamos a incidir aquí sobre el problema de la reproducción y venta ilegal de programas comerciales, pero sí los vamos a comentar bajo el punto de vista de su lectura por el cassette.

Las grabaciones "piratas", están en su inmensa mayoría realizadas utilizando una cinta original. Se sitúa ésta en un cassette y una cinta virgen de la peor calidad (léase más barata) o bien en un cassette con doble pletina y uno graba y otro lee. Así de sencillo. Pero los resultados no son nada buenos en calidad y mucho menos si el original es a su vez otra cinta pirata (copias de copia).

Cada vez que se produce una copia por estos medios caseros, se introducen una serie de ruidos que distorsionan la información, lo cual conduce a la producción de errores de lectura, por mala interpretación de la misma.

Toda esta exposición viene en favor del cassette de Amstrad y de las cintas originales, concebidas para él. He observado multitud de veces cómo cunde el desánimo entre los usuarios cuando detecta estos errores que son achacado al ordenador, mientras el verdadero culpable es la cinta mala y peor grabada.

A pesar de lo dicho y cuando se utilizan cassettes externas (664 y 6128), aun a pesar de ser cintas originales, pueden presentarse problemas del azimut de la cabeza en estos casos. Al final del capítulo indicamos cómo subsanarlos.

Tras este inciso, continuemos con la forma de grabar la información en la cinta. Las grabaciones en el Amstrad, no se reducen sólo a las estrictas (datos), sino que incorporan informaciones complementarias, conducentes a optimizar dichas grabaciones desde el punto de vista de su calidad e información sobre las mismas, con vistas a una buena reproducción.

Toda grabación de programa o fichero, va precedida de una cabecera, conocida **bloque de cabecera**, que básicamente recoge la siguiente información:

GAP	**Registro de CABECERA**	**Registro de DATOS**

Veamos cada una de estas partes.

GAP (hueco): Es un espacio entre bloques consecutivos a fin de diferenciarlos. Si pone en marcha una cinta para su lectura, podrá "percibir los **silencios**" que el GAP produce.

CABECERA: Está formado por un conjunto de 63 bytes, cuyas funciones asignadas son las siguientes:

Bytes	Contenido
0-15	Nombre del fichero
16	Número de bloque
17	Señal de último bloque (último <> 0)
18	Tipo de fichero
19-20	Longitud de los datos
21-22	Localización de datos en memoria.
23	Señal primer bloque (primero <> 0)
24-25	Longitud total del fichero en bytes
26-27	Dirección de ejecución (programa en Código Máquina)
28-63	No utilizados (disponibles para el usuario)

El byte 18 varía su valor en la forma siguiente:

bit 0	Protección
bits 1 a 3	000 Programa BASIC 001 Programa Binario 010 Pantalla 011 ASCII
bits 4 a 7	Versión

DATOS: Este registro está dividido en un máximo de 8 partes. Cada parte contiene 256 bytes, lo cual hace un total de 2048 bytes. Como 1024 bytes = 1K, quiere decirse que cada bloque contiene 2K de información (ver apartado correspondiente de este mismo epígrafe).

Estos ocho segmentos son precedidos, por uno especial, denominado **segmento de cabecera,** cuyo contenido es el siguiente:

GAP	**2048 bits**	**bit a 0**	**byte de sincronismo**

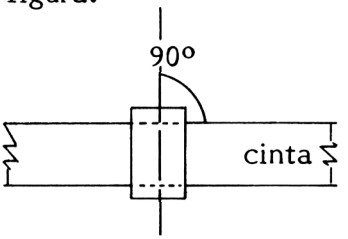
Donde el GAP separa los bloques. Los 2048 bits son para poner de acuerdo las velocidades de grabación y lectura de datos y el byte de sincronismo homogeiniza la lectura, a fin de no perder la sincronía.

Por último, los segmentos que, como hemos visto, están formados por 256 bytes en serie, llevan además 2 bytes adicionales, denominados CRC (Código Redundante Cíclico), encargados de la detección de errores.

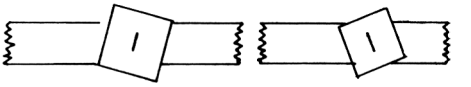
El azimut de la cabeza lectora/grabadora

La pieza fundamental de un cassette respecto a la calidad de grabación y lectura es, sin duda, la **cabeza magnética,** que en esencia es un electroiman minúsculo, capaz de magnetizar las partículas de material magnético contenido en el soporte de cinta plástica y que constituyen la cinta cassette.

La cabeza magnética está alojada en un hueco de la cabeza metálica que todos conocemos, sobre la que se desliza la cinta. Pues bien, este hueco debería estar alineado de tal forma, que el ángulo con la cinta fuese de 90º, tal como se indica en la figura.



Eje del hueco
de la cabeza magnética
CABEZA BIEN ALINEADA



CABEZAS MAL ALINEADAS

Pero este ángulo rara vez se encuentra ajustado con exactitud y, mucho menos, en cassettes muy usados y a veces mal usados.

Si un programa fue grabado con una alineación perfecta de la cabeza, y se intenta leer con otro muy desajustado, se producirán, sin duda, errores. Igual ocurrirá a la inversa, si fue grabado en uno mal ajustado y se intenta leer con uno ajustado, también dará error.

Esta probabilidad se hace muy alta cuando se dan simultáneamente en la grabación las siguientes condiciones: cabeza desajustada, ruidos intermedios por copia defectuosa (pirata) y tono general bajo en la grabación a excesivamente alto.

Otro origen de errores puede ser la suciedad de la cabeza (cassettes antiguos y muy usados). En este caso, basta una limpieza con un producto adecuado al efecto, y de venta en establecimientos de música.

No osbtante, puede variarse el ángulo de la cabeza, consiguiendo de esta forma ajustarlo, a fin de que puedan leerse programas que antes no "entraban".

Esta manipulación se la aconsejamos únicamente para cassettes externos y no para el incorporado en el CPC 464. Siga atentamente las indicaciones que le hacemos a continuación. Ayudado por los dibujos y con un poco de práctica le acabarán entrando todos los programas "que estén bien grabados".

Método para alinear la cabeza del cassette

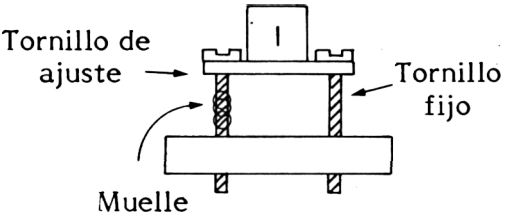
1. Introduzca la cinta con el programa.

2. Retire las conexiones del cassette, REM y EAR.

3. Pulse PLAY en el cassette y coloque el volumen más bajo de lo habitual, ajustándolo de forma que se oiga bien, sin molestias.

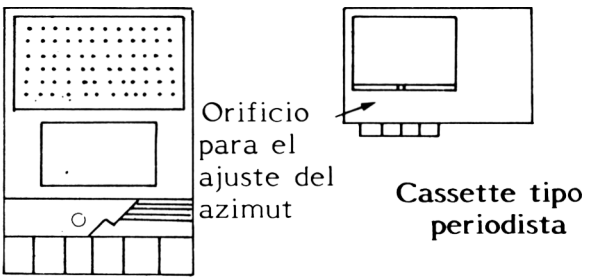
4. Si tiene ajuste de tono, póngalo en agudos (máx. tiple).

5. Coloque un destornillador de punta fina (relojero) en el tornillo de ajuste de la cabeza que se indica en la figura.



Casi todos los cassettes disponen de un agujero o ranura al efecto. En caso contrario puede hacerse con un taladro pequeño o clavo de acero al rojo -si se trata de plástico- cuidando de que caiga en la vertical del tornillo, cuando la cabeza está en PLAY, no en reposo.

Existen muchos modelos, con una chapa de aluminio muy fina, en la que suelen estar grabados los nombres de las teclas. Casi con toda seguridad debajo existe el mencionado orificio. Levante con cuidado la chapa que va pegada y cuando descubra el orificio, haga otro en la chapa y vuelva a pegarla. Quizá no le quede muy bonito, pero desde luego sí eficaz.



6. Muevan despacio el tornillo hacia un lado tratando de escuchar un aumento de los agudos. Si ocurre al revés, es decir se oye más grave, invierta el sentido de giro.

   Suele ser suficiente un movimiento de 1/4 a 1/2 vuelta o quizá 3/4 de vuelta. Si da varias vueltas puede llegar a sacar el tornillo completamente y su muelle, lo cual le obligará a desmontar toda la carcasa para roscarlo de nuevo. Hágalo pues con precaución.

7. Una vez detenido el giro en el máximo agudo, coloque las conexiones, rebobine y haga RUN", para cargar el programa.

   Si no lo consigue, inténtelo de nuevo afinando el oido. Si a pesar de ello, no lo consigue deseche la grabación.

8. Tras estas manipulaciones, puede que los programas grabados por Vd. con anterioridad, no entren por culpa del cambio efectuado en la alineación. Aplicando el mismo procedimiento, con una de sus cintas, puede restituir sensiblemente las condiciones iniciales.

4. Ficheros en disco

Generalidades

Los ficheros secuenciales en disco tienen las mismas características que en cinta, como consecuencia de su propia concepción. Sin embargo, su manejo resulta mucho menos rígido en el sentido de que el acceso al comienzo del fichero, al menos es directo, a través del directorio del disco.

El acceso a un determinado dato, como vimos anteriormente, obliga a la lectura de todos los anteriores, es decir, si deseamos "leer" el dato número N, el proceso será abrir el fichero e ir leyendo los datos 1, 2, 3..., N. Veamos una rutina al efecto, que lee un dato determinado del fichero "Nombres", utilizado en el capítulo de ordenación.

```
10 ' LECTURA DE UN ELEMENTO
20 ' ........ FS7 .........
30 OPENIN "NOMBRES"
40 INPUT "NUMERO DE ORDEN BUSCADO ";N
```

```
50 FOR K=1 TO N
60 INPUT #9,N$
70 NEXT K
80 CLOSEIN
90 PRINT "EL NOMBRE No.";N;" ES ";N$
```

El procedimiento es válido, pero implica el conocimiento exacto de la posición del dato a buscar. Es muy frecuenta tratar de encontrar un nombre o parte de él sin saber su posición. Este problema de búsqueda es análogo al planteado en el capítulo de matrices y, por tanto, pueden extrapolarse los métodos expuestos en él, al manejo de archivos.

Veamos un ejemplo. Se trata de buscar la cadena "PEPE" en un fichero. En el programa que sigue se ha utilizado el método de **búsqueda por igualdad,** aunque podía utilizarse el de inclusión, más potente.

```
10 ' BUSQUEDA DE UN NOMBRE
20 ' ........ FS8 ........
30 INPUT "NOMBRE BUSCADO ";N$
40 OPENIN "NOMBRES"
50 WHILE NOT EOF
60 K=K+1
70 INPUT #9,A$
80 IF N$=A$ THEN 120
90 WEND
100 PRINT "NO EXISTE EL NOMBRE ";N$
110 END
120 PRINT "EL NOMBRE ";N$;" OCUPA EL LUGAR ";K
```

¿Qué ocurre si el fichero contiene, además de nombres, otros datos asociados al mismo? Veamos un ejemplo. Se han introducido en un fichero los nombres de varios amigos, acompañados de su dirección y teléfono, de tal forma, que se encuentran grabados secuencialmente en la forma:

Dato Nº

1	2	3	4	5	6	7	8
Nombre 1	Direc. 1	Tel. 1	Nombre 2	Tel. 2	Nombre 3		

Contenido

La búsqueda de nombres habrá de tener en cuenta que éstos se encuentran en los lugares 1, 4, 7, 10, etc., de manera que la rutina anterior debe ser modificada. Veamos como quedaría.

```
10 ' BUSQUEDA DE UN NOMBRE
15 ' Registros con varios campos
20 ' ........ FS9 ........
30 INPUT "NOMBRE BUSCADO ";N$
40 OPENIN "NOMBRES"
50 WHILE NOT EOF
60 K=K+1
70 INPUT #9,A$(0),A$(1),A$(2)
80 IF N$=A$(0) THEN 120
90 WEND
100 PRINT "NO EXISTE EL NOMBRE ";N$
110 END
120 PRINT "NOMBRE    ";A$(0)
130 PRINT "DIRECCION ";A$(1)
140 PRINT "TELEFONO  ";A$(2)
```

La línea 70 ya no lee sólo un dato, sino 3: nombre, dirección y teléfono, almacenados en una matriz, que no necesita dimensionamiento (sólo tiene 3 elementos). La salida de datos no es como en el caso anterior el lugar que ocupa, sino el nombre buscado, asociado a la dirección y teléfono que le corresponden.

En el apéndice a este capítulo se incluye un programa completo de agenda, válido para disco y cinta, al cual nos remitimos, a fin de profundizar en todos los temas tratados, a la vez que constituye un programa de utilidad.

Borrar ficheros del disco

Cuando se crea un fichero en un disco, éste se incluye en su directorio. Muchas veces los ficheros grabados originalmente, son manipulados y grabados de nuevo con las modificaciones habidas, utilizando el mismo nombre que el original.

¿Qué ocurre en este caso? El controlador del disco al intentar grabar el nuevo fichero, observa que existe uno con el mismo nombre y como no pueden coexistir ambos en el directorio, procede a etiquetar el antiguo, como tipo .BAK, y al nuevo con tipo que se le indica o nada, por omisión.

De esta forma pueden coexistir fichero "viejo" y "nuevo", identificados con un nombre común, pero tipo distinto.

Por ejemplo, si se grabó un fichero con el nombre:

DATOS 1.AL

y tras modificarlo se vuelve a grabar con idéntico nombre, en el disco existirán dos ficheros

DATOS 1.BAK	(viejo)
DATOS 1.AL	(nuevo)

Si se repite la operación se perdería el original el segundo pasaría a tipo BAK y el último con el nombre completo, y así sucesivamente:

~~DATOS 1.BAK~~	(viejo)
DATOS 1.BAK	(nuevo)
DATOS 1.AL	(novísimo)

Esto resulta de gran utilidad, pues nos permite -de forma automática- disponer de una **copia de seguridad,** aunque esté un poco anticuada. En muchas ocasiones, resulta que las modificaciones no están bien hechas y gracias a ésto podemos recuperar la versión anterior del fichero.

A pesar de todo lo dicho, la "ley de conservación de ficheros" no es eterna. Queremos decir, que también en ocasiones tenemos un disco lleno de ficheros y programas del tipo .BAK, completamente inútiles y ocupando sitio.

Veamos cómo proceder al borrado de todos estos ficheros residuales.

Como conoce sin duda, la orden Basic de borrado de programas en disco es ¦ ERA. Esta orden es también válida para los ficheros, sean del tipo que sean. A fin de cuentas para el ordenador todo son ficheros, distinguidos por el tipo.

Si el borrado quiere hacerse extensivo a todos los ficheros del tipo .BAK, lo más cómodo es utilizar el símbolo comodín *, que viene a significar "todos". La orden es:

¦ ERA, "*.BAK

Es decir, "Borra todos los ficheros, cualesquiera que sean sus nombres y que respondan al tipo BAK".

Si la operación de borrado debe actuar sobre un fichero concreto, debe indicarse su nombre y tipo, por ejemplo:

```
|ERA,"DATOS.AL
|ERA,"DEL.BAS
```

Cuando se manejan ficheros grandes, por ejemplo, de 100 K, resulta que entre el programa que lo maneja y el fichero, no queda sitio para la aparición de un BAK en la misma cara, lo cual obliga a cambiar de cara o disco, cada vez que se producen modificaciones en el mismo y sus consecuentes grabaciones.

Este problema tiene una sencilla solución: basta con incluir una orden de borrado, previa a la grabación. Por ejemplo:

```
1000 ' RUTINA DE BORRADO-GRABACION
1005 ' ........... F10 ............
1010 |ERA,"DATOS.AL"
1020 OPENOUT "DATOS.AL"
1030 FOR K= 1 TO 100
1040 WRITE #9,A$(K)
1050 NEXT K
1060 CLOSEOUT
```

La línea 1010 libera el espacio del disco ocupado por "DATOS.AL" para luego ser ocupado por el nuevo fichero del mismo nombre.

Si necesita utilizar este sistema por la falta de espacio en el disco, es conveniente que guarde alguna copia del fichero en otro disco o cinta si lo prefiere. Duplicar todos los archivos y máxime cuando son grandes, es una regla que le producirá grandes satisfacciones; el caso contrario, sin duda, quebraderos de cabeza.

Listar el contenido de un fichero

No nos referimos en este apartado a la operación de lectura de ficheros acompañada de PRINT en pantalla e/o impresora -tal como se describió en este mismo capítulo- sino a volcar el contenido de un fichero en pantalla o impresora, sin tener que recurrir a un programa o rutina al efecto.

Se trata de aprovechar la facilidad del programa TYPE del sistema operativo CP/M, con que cuenta su ordenador. Para ello, proceda en la forma siguiente:

1. Inserte la cara 1 de CP/M del disco de sistema, o mejor, una copia de seguridad del mismo y teclee ¦ CP/M para llamar a dicho sistema operativo.
2. Una vez tenga en pantalla la impronta A>, teclee TYPE (en mayúsculas o minúsculas, es lo mismo) seguido de ENTER o RETURN.
3. Aparecerá el mensaje "Enter file:", introduzca el nombre completo del fichero a listar (nombre y tipo) y antes de presionar ENTER, cambie el disco de sistema por el que contiene el fichero.
4. Si el fichero no existe o su nombre se tecleó incorrectamente, aparecerá el mensaje:

   ERROR: Not Typed: nombre del fichero. No file

   Indicando la imposibilidad del listado, puesto que no existe ese fichero.

   En caso contrario obtendrá en pantalla el contenido del fichero.
5. Si en vez de la pantalla desea utilizar la impresora, debe teclear entre los pasos 1 y 2: < CONTROL > P. De esta forma el listado se dirige a la impresora.

   Para volver a listar sólo por pantalla, basta con teclear de nuevo < CONTROL > P, que actúa como un ON/OFF de la impresora.

5. Apéndice al capítulo

El programa que presentamos a continuación, permite construir una agenda de direcciones y teléfonos, con los datos archivados en cinta (o disco). Por tanto, es directamente ejecutable por el Amstrad CPC464.

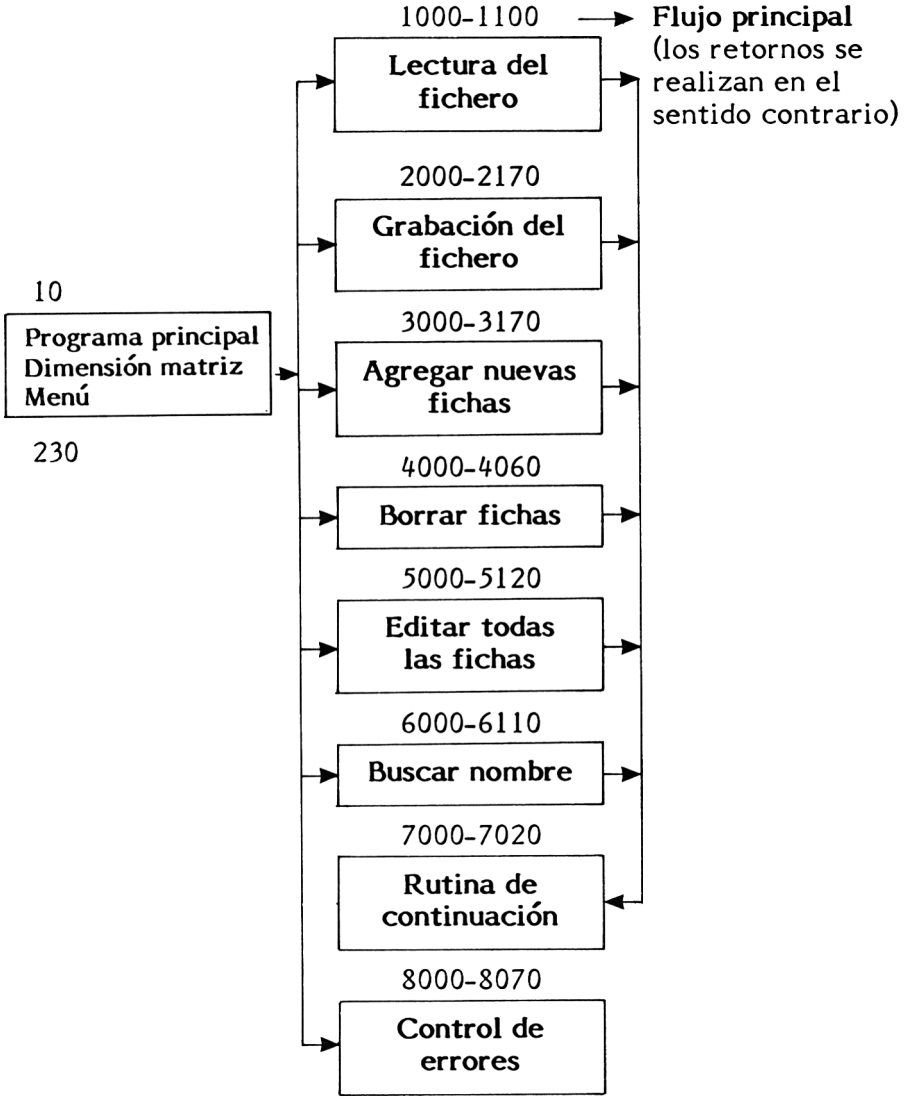
Para los modelos CPC664 y 6128 también es válido el programa tal como está, aunque sobran las líneas 2020 a 2050, pero la grabación se hará en el disco. Para utilizar la cassette adicional en estos modelos, incluya la siguiente línea:

```
25 ¦TAPE
```

y el programa funcionará exactamente igual que en un 464.

El programa está organizado como un pequeño programa de entrada de datos generales y menú de selección, y 8 subrutinas manejadas desde dicho menú.

El organigrama básico es el siguiente:



```
5 '  ****************************************
10 ' *** A G E N D A DIRECCIONES Y TELEFONOS ***
20 ' ****   Especial para archivo en cinta  ****
25 ' ****************************************
30 ON ERROR GOTO 8000
40 MODE 1
50 'Pantalla para mensajes
60 WINDOW #1,1,40,24,25
70 PEN #1,0:PAPER #1,1
80 ' Matriz de datos (max 100 fichas)
90 N=100
100 DIM FI$(N,6)
120 CLS:' **********  M E N U  *************
130 LOCATE 12,3: PRINT "M E N U"
140 LOCATE 6,5: PRINT "1. LEER EL FICHERO"
150 LOCATE 6,7: PRINT "2. GRABAR EL FICHERO"
160 LOCATE 6,9: PRINT "3. AGREGAR FICHA"
170 LOCATE 6,11: PRINT "4. BORRAR FICHA"
180 LOCATE 6,13: PRINT "5. VER FICHAS"
190 LOCATE 6,15: PRINT "6. BUSCAR NOMBRES"
195 PRINT #1,"ESCRIBIR SIEMPRE CON MAYUSCULAS"
200 INPUT #1," SELECCIONE OPCION ";OP
210 CLS#1:IF OP<1 OR OP>6 THEN PRINT CHR$(7):GOTO 200
220 ON OP GOSUB 1000,2000,3000,4000,5000,6000
230 GOTO 120
1000 ' ***********  LEER EL FICHERO
1010 CLS:INPUT #1,"NOMBRE DEL FICHERO ";F$
1020 OPENIN F$
1030 FOR FICHA=1 TO N
1040 FOR CAMPO=0 TO 6
1050 INPUT #9,FI$(FICHA,CAMPO)
1060 NEXT CAMPO
1070 IF EOF THEN 1090
1080 NEXT FICHA
1090 PRINT #1,"FICHERO ";F$;" CARGADO"
1100 GOSUB 7000:RETURN
2000 ' ***********  GRABACION DE LOS DATOS
2010 CLS:INPUT #1,"NOMBRE DEL FICHERO";F$
2020 INPUT #1,"VELOCIDAD (0=SEGURO 1=RAPIDO)  ";V
2030 IF V=0  THEN SPEED WRITE 0:GOTO 2060
2040 IF V=1  THEN SPEED WRITE 1:GOTO 2060
2050 GOTO 2020
2060 CLS:PRINT CHR$(27);".  G R A B A N D O    ";CHR$(27)
2070 OPENOUT F$
2080 FOR FICHA=1 TO N
2090 IF FI$(FICHA,0)="" THEN 2140
2100 FOR CAMPO=0 TO 6
2110 PRINT #1,FI$(FICHA,CAMPO)
2120 WRITE #9,FI$(FICHA,CAMPO)
```

```
2130 NEXT CAMPO
2140 NEXT FICHA
2150 CLOSEOUT
2160 PRINT #1,"GRABACION TERMINADA"
2170 GOSUB 7000:RETURN
3000 ' ***********  AGREGAR FICHAS NUEVAS
3010 CLS
3020 FOR FICHA =1 TO N
3030 IF FI$(FICHA,0)="" THEN 3060
3040 NEXT FICHA
3050 PRINT #1,"NO HAY MAS FICHAS DISPONIBLES":GOSUB 7000:RETURN
3060 CLS:RESTORE 8080
3070 FI$(FICHA,0)="1"
3080 FOR CAMPO=1 TO 6
3090 READ A$:PRINT A$+": ";
3100 INPUT FI$(FICHA,CAMPO)
3110 NEXT CAMPO
3120 CLS #1:INPUT #1,"DATOS CORRECTOS S/N)";Z$
3130 IF Z$="S" THEN 3140 ELSE 3060
3140 CLS #1:INPUT #1,"INTRODUCIR MAS FICHAS (S/N)";Z$
3150 IF Z$="S" THEN 3000
3160 IF Z$="N" THEN RETURN
3170 GOTO 3140
4000 ' ***********  BORRAR UN FICHA
4010 CLS:INPUT #1,BORRAR LA FICHA No. ":FICHA
4020 INPUT #1,"ESTA SEGURO (S/N)";Z$
4030 IF Z$="S" THEN FI$(FICHA,0)="":GOTO 4060
4040 IF Z$="N" THEN 4060
4050 GOTO 4020
4060 GOSUB 7000:RETURN
5000 ' ***********  EDITAR TODAS LAS FICHAS
5010 CLS
5020 FOR FICHA=1 TO N
5030 IF FI$(FICHA,0)="" THEN 5110
5040 CLS:PRINT "FICHA No.";FICHA;")"
5050 FOR CAMPO=1 TO 6
5060 PRINT FI$(FICHA,CAMPO)
5070 NEXT CAMPO
5080 PRINT #1,"PRESIONE UNA TECLA PARA CONTINUAR       <Z> PARA PARAR
5090 Z$=INKEY$:IF Z$="" THEN 5090
5100 IF Z$="Z" THEN 5120
5110 NEXT FICHA
5120 RETURN
6000 ' BUSCAR NOMBRES
6010 CLS:INPUT #1,"NOMBRE A BUSCAR";N$
6020 FOR FICHA=1 TO N
6025 RESTORE
6030 IF INSTR(FI$(FICHA,1),N$)=0 THEN 6080
6040 FOR CAMPO=1 TO 6
```

```
6045 READ A$:PRINT A$;":";
6050 PRINT FI$(FICHA,CAMPO)
6060 NEXT CAMPO
6070 GOSUB 7000
6080 NEXT FICHA
6090 PRINT #1,"NO HAY MAS FICHAS"
6100 GOSUB 7000
6110 RETURN
7000 ' ********** RUTINA DE CONTINUACION
7010 PRINT #1,"PRESIONE UNA TECLA PARA CONTINUAR":CALL &BB18
7020 RETURN
8000 ' ********** CONTROL DE ERRORES
8010 CLS #1
8020 IF ERR=32 THEN PRINT #1,"ERROR EN EL MANEJO DEL DISCO No";DERR:GOTO 8040
8030 PRINT #1," ERROR No.";ERR;" EN LA LINEA ";ERL
8040 GOSUB 7000
8050 PRINT #1,"VOLVEMOS AL MENU"
8060 GOSUB 7000
8070 GOTO 120
8080 DATA "Nombre  ","Calle   ","Ciudad  ","Provinc.","Cod.pos.","Telefono"
```

A continuación comentaremos brevemente cada una de las partes del programa, resaltando las líneas más importantes.

*** Programa principal**

30 — Controla los posibles errores direccionando a la subrutina 8000 para dar información sobre los mismos (ver lista de errores en el manual).

60-70 — Establece una ventana para mensajes, en vídeo inverso, de dos líneas: la 24 y la 25.

90-100 — Dimensiona una matriz bidimensional de 100 fichas (registros) y cada ficha con 6 datos (campos), que corresponden a la siguiente estructura de ficha:

Nombre:	FI$(Ficha, 1)
Calle:	FI$(Ficha, 2)
Ciudad:	FI$(Ficha, 3)
Provincia:	FI$(Ficha, 4)
Cód. Postal:	FI$(Ficha, 5)
Teléfono:	FI$(Ficha, 6)

También se utiliza el elemento 0 de la matriz FI$ (Ficha, 0), para el control de cada ficha, en el sentido de estar vacía o llena.

Ficha vacía ⟹ FI$(FICHA, 0) = ""
Ficha llena ⟹ FI$(FICHA, 0) = "1"

Así, para realizar el borrado de una ficha es suficiente hacer FI$(FICHA, 0) = "". Por el contrario, si se crea una nueva se iguala a "1".

120-190 Presentacion del menú.

200-230 Entrada de la opción y direccionamiento múltiple a la subrutina correspondiente, mediante ON GOSUB. La línea 230 recoge los retornos de dichas subrutinas y devuelve el control al menú.

*** Subrutina 1000. Lectura del fichero**

1010-1020 Introducen el nombre del fichero y lo abren para leer.

1030-1080 Bucle principal de lectura de fichas.

1040-1060 Bucle interno de lectura de los datos de una ficha (campos).

1070 Esta línea se encarga de controlar la señal de "fin de fichero". Si se detecta dicha señal, se abandona el bucle principal (todas las fichas del archivo han sido leídas).

1090-1100 Indica por la ventana creada, que el fichero ha sido leído y direcciona a la subrutina 7000 para proseguir. Por último, retorno al programa principal.

* **Subrutina 2000. Grabar el fichero en cinta/disco**

El funcionamiento es semejante al anterior. En el sentido contrario, los datos son almacenados en vez de leídos.

La línea 2090 controla aquellas fichas vacías, a fin de que no sean guardadas en el archivo, de esta forma se reduce el tamaño del fichero al estricto.

Tras el cierre del fichero se utiliza la subrutina 7000 para continuar y se retorna al programa principal.

* **Subrutina 3000. Agregar fichas**

Un fichero de este tipo ha de ser dinámico, es decir, las fichas almacenadas deben poder ser modificadas o borradas en cualquier momento, asimismo, debe ser posible añadir más dentro del límite impuesto por el dimensionamiento de la matriz (90-100).

Esta subrutina, se encarga precisamente de esta última tarea. En primer lugar, se busca un hueco en la lista. Aquí cabe hacer las siguientes observaciones:

- Si la lista **es compacta** (sin huecos), el primer hueco será el situado después de la última ficha llena.
- Si la lista **tiene huecos** que provienen del borrado, se buscará desde el principio la ubicación de dicho hueco con el fin de aprovecharlo.

Por ejemplo, en las listas que ofrecemos a continuación puede verse que la primera contiene 5 fichas con datos seguidos, de forma que una nueva ficha se ubicará en la posición 6ª.

Sin embargo, la segunda lista contiene huecos en tercer y cuarto lugar, por tanto, la nueva ficha se ubicará en el lugar tercero. Si introdujésemos más fichas irían a los lugares 4, 6, 7... etc.

Para permitir la corrección de los datos introducidos se utilizan las líneas 3120-3130. Piense que la corrección de un dato de una ficha, no está contemplada, por lo que corregir un dato de un campo erróneo, supone volver a escribir la ficha completa. No resulta difícil añadir esta nueva posibilidad. Sería de gran provecho para el lector intentarlo. En el programa no se ha realizado, para darle mayor sencillez de comprensión.

1. LISTA COMPACTA		2. LISTA CON HUECOS	
Ficha nº	Valores de FI$(FICHA,0)	Ficha nº	Valores de FI$(FICHA,0)
1	"1"	1	"1"
2	"1"	2	"1"
3	"1"	3	""
4	"1"	4	""
5	"1"	5	"1"
6	""	6	""
7	""	7	""
8	""	8	""
.	.	.	.
.	.	.	.
.	.	.	.

*** Subrutina 4000. Borrar una ficha**

Como hemos comentado anteriormente, la diferencia entre una ficha llena y una vacía es que el valor de FI$(FICHA, 0) es "1" y "", respectivamente.

Por tanto, una vez conocido el número de la ficha a borrar por la línea 4010, se procede a igualar a "nada", dicha variable de la matriz.

Como el borrado siempre es problemático, en el sentido de que con demasiada frecuencia borramos lo que no deseamos, antes de "destruir" la ficha, la línea 4020 le pregunta si está seguro de su drástica resolución.

El final de la rutina es común a todas las anteriores, se direccionan a la subrutina 7000 de continuación y después se retorna al programa principal (menú).

*** Subrutina 5000. Editar todas las fichas**

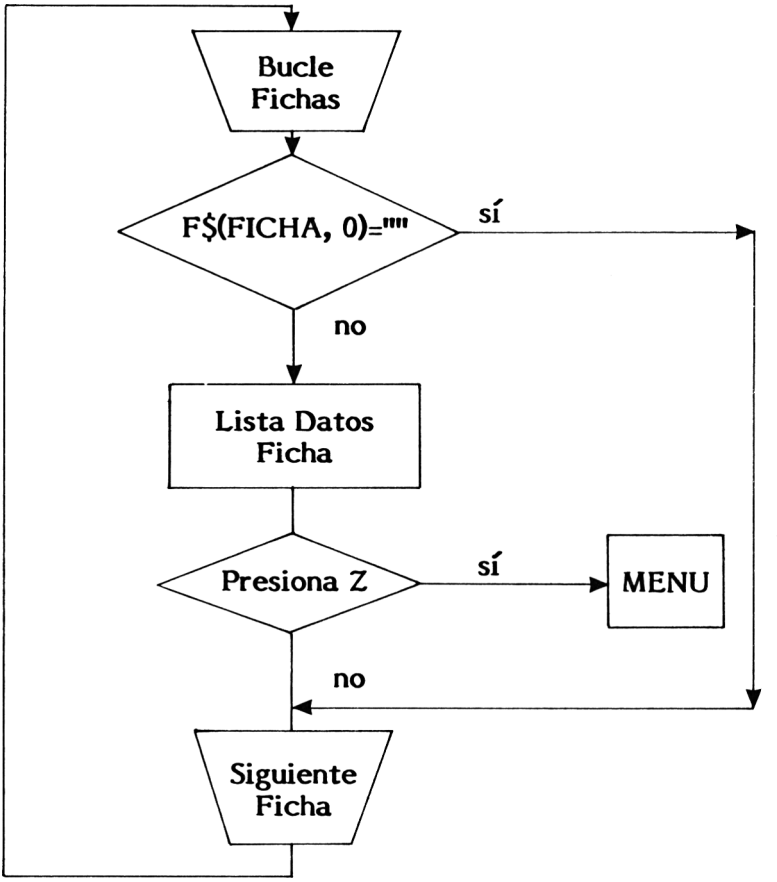
Permite observar en pantalla todas las fichas almacenadas, con todos sus datos.

El método es simple. Un bucle exterior que controla el número de fichas (5020) y otros interno (5050) que controla los 6 campos, más el 0, son los encargados de manejar una a una las fichas.

Con el fin de no mostrar los huecos, se ha incluido la línea 5030 que produce un salto a la siguiente ficha si la actual está vacía o se borró previamente. Piense que los valores de FI$(FICHA, 0) constituyen un directorio de fichas útiles.

Por otro lado, a veces se quieren ver varias fichas, pero no todas. Para ello, se ha construido el control de las líneas 5080 a 5100, de forma que si se presiona una tecla cualquiera (a excepción de Z) continúa listando la siguiente ficha; pero si se presiona "Z" se da por terminada la tarea, retornando al menú.

El esquema básico de esta rutina es el siguiente:



*** Subrutina 6000. Buscar nombres**

Es, sin duda, la rutina más útil para el trabajo. La anterior permite la obtención de los datos de una ficha, pero de forma secuencial, es decir, a base de leer todas las anteriores.

Esta opción permite el acceso directo completo a una ficha determinada, con la condición de conocer el nombre del primer campo, o parte de él. Esto es posible gracias a la instrucción INSTR incluida en el BASIC del Amstrad.

La instrucción INSTR, se articula en la siguiente forma:

INSTR (K, "cadena en la que se busca", "subcadena buscada")

siendo **K** un parámetro opcional que indica la posición del carácter de la cadena en la que busca, a partir de la cual se iniciará dicha búsqueda. Por omisión se toma el primer carácter. INSTR da el valor 0 si la subcadena buscada no está contenida en la cadena en la cual se busca, 0 un valor númerico que indica la posición a partir de la cual se produce la coincidencia.

Esta instrucción es de gran efectividad en la búsqueda de datos. Veamos con un ejemplo su funcionamiento práctico en nuestro caso. Imaginemos que la ficha 24 contiene en su primer campo:

FI$(24,1) = "VICENTE LOPEZ CUEVAS"

Si buscamos la subcadena N$="LOPEZ", se verificará que:

INSTR (FI$(24,1), N$) <> 0 ⟶ Está contenida

mientras que si la subcadena fuese N$="VIRIATO", se verificará que:

INSTR (FI$(24,1), N$) = 0 ⟶ No está contenida

Volviendo a la rutina, veremos que el bucle de la línea 6020 es el encargado de buscar en todas las fichas.

Si la condición de la línea 6030 se verifica, quiere decir que no existe la subcadena en esa ficha y, por tanto, salta al NEXT a fin de continuar la búsqueda.

En caso contrario, la habrá encontrado y el bucle interno de la línea 6040 a 6060, se encarga de mostrarnos todos los datos de esa ficha.

Observe que a pesar de haberse encontrado una ficha con el nombre que buscamos, el programa se encuentra con un NEXT en la 6080 que le obliga a seguir buscando más fichas. Esto se debe a que en una agenda pueden, evidentemente, repetirse nombres y apellidos y la primera ficha encontrada puede o no ser la que buscamos.

Sólo se ha incluido la posibilidad de buscar por un campo el nombre, pero sería fácil realizar la búsqueda por cualquier campo.

* Subrutina 7000. Continuación

Esta subrutina es tomada desde todos los anteriores en su fase final. Un mensaje en la ventana #1 nos indica que ha terminado la tarea en curso y que para continuar debemos presionar cualquier tecla.

La pausa indefinida producida por esta subrutina, permite visualizar tanto tiempo como deseemos, el contenido de la pantalla. Dicha pausa se obtiene llamando a una rutina de la ROM (CALL &BB18) ya que el Amstrad no dispone de una sentencia BASIC que produzca este efecto como lo hace PAUSE 0, por ejemplo, en el Spectrum.

* Subrutina 8000. Control de errores

Sin duda la utilización del programa puede dar lugar a errores debidos a múltiples factores: entradas de datos anómalas, designaciones de ficheros incorrectas o inexistentes, etc. Por todo ello, mediante la línea 30 se le indica al ordenador que en caso de producirse un error, se traslade el control a esta rutina.

Puesto que el programa, como se ha indicado anteriormente, puede trabajar además de con cinta, con discos, la subrutina emite dos tipos de mensajes.

El primero (línea 8030) nos indica el número de error cometido. El segundo, en caso de ser un error de disco, indica además el subtipo.

Todo esto se realizará ayudado de las funciones ERR, ERL y DERR:

ERL Indica el número de línea en que se produce el error.
ERR Contiene el código del error que se ha producido.
DERR Contiene el código de error del tipo disco, catalogados genéricamente por ERR como 32.

Para más información, vea en el apéndice final los errores y sus códigos de identificación.

Dado que la subrutina 8000 no resulta estrictamente necesaria en el programa puede suprimirla, si lo desea, eliminando también la línea 30.

CAPITULO 5

FICHEROS DE ACCESO DIRECTO

1. **Introducción**

2. **Conceptos previos**

3. **El programa RANDOM FILES**
   * RANDOM-F.BAS
   * RANDOM.BIN

4. **Apéndice al capítulo (Programa Agenda)**
   * Preliminares
   * Descripción del programa

Ficheros de Acceso Directo

1. Introducción

Hemos visto en el capítulo anterior cómo en los ficheros secuenciales -para leer un registro- se ha de proceder a la lectura de todos los anteriores; y cómo -para corregir datos- hemos de cargar todo el fichero, modificarlo y grabar el nuevo fichero con la consiguiente aparición de uno del tipo BAK, etc.

Todos estos "inconvenientes" se subsanan con los ficheros de **acceso directo** -también conocidos como **aleatorios.** En éstos, el acceso a un dato o grupo de datos es inmediato y directo, sin pasar por los anteriores, tanto para lectura como para grabación, de tal forma que la corrección de un registro no exige la manipulación del resto ni la alteración del fichero en el disco, tomado como conjunto.

Desgraciadamente, en la serie CPC de Amstrad no estaba prevista la utilización de ficheros aleatorios desde el Locomotive BASIC, y ésta es la razón por la cual no existen órdenes ni comandos apropiados.

Esta deficiencia se ha tratado de subsanar a través de un programa **-Random Files-** incluido en el disco de regalo, junto con otros programas de utilidad.

2. Conceptos previos

El ejemplo más claro del significado del almacenamiento de acceso directo, nos lo ofrece el propio sistema de gestión del disco.

Cuando formateamos un **disco,** vemos que éste se divide en 40 **pistas** (círculos concéntricos) y que cada pista se divide a su vez en 9 partes iguales, que reciben el nombre de **sectores.**

La capacidad de un sector es de 512 **bytes,** por lo que, realizando una pequeña operación (40 · 9 · 512 = 184.320), llegamos a la conclusión de que cada cara de un disco dispone de 184.320 bytes; lo cual supone 180K de capacidad máxima (184.320/1.024 = 180K).

Cuando se graba un programa, se incluyen en el directorio -además del nombre- la pista y sector donde comienza y donde acaba. Tenga en cuenta que no tienen por qué ser contiguos y, de hecho, lo normal es que estén separados, de manera que un fichero se encuentra repartido por varios sectores del disco.

Al recibir una orden de lectura, se accede, en primer lugar, al directorio y, si existe el fichero, se recoge la información sobre la forma en que dicho fichero está repartido por el disco. Esto constituye un acceso directo, puesto que no hay que leer ficheros anteriores para acceder al buscado.

La antítesis de lo expuesto es la cinta -almacenamiento **secuencial**- donde para leer un fichero (programa) hay que comenzar desde el principio. Por supuesto que Vd. puede contar las vueltas y saber más o menos dónde comienza, pero la maniobra manual de hacer avanzar la cinta sigue constituyendo una búsqueda secuencial (¡y encima "a mano"!).

Cuando se crea un fichero de acceso directo también se crea un índice, que permite localizar cualquier registro; de manera que existe una analogía entre ficheros de acceso directo y disco en cuanto a la forma de guardar la información.

	Disco	**Fichero Acceso Directo**
Directorio de	Ficheros	Registros
Identificación mediante	Nombre del fichero	Número del registro
Organización	Sectores de 512 bytes	Registros de igual longitud

La identificación de un fichero viene dada por su nombre, mientras que la de un registro viene dada por su número de orden. El conocer su **número de orden** es importante, pues permite el acceso directo propiamente dicho. Esto no quiere decir que sea estrictamente necesario, pues siempre podemos buscar un elemento recurriendo las técnicas utilizadas en los de acceso secuencial, pero evidentemente perdemos su mayor ventaja.

Una particularidad es la longitud de cada registro. En los secuenciales podíamos tener la información almacenada, una tras otra, con longitudes distintas. Por ejemplo:

"LUIS", "JOSE MANUEL", "PEDRO"
4 11 5

Sin embargo, los registros de un fichero de acceso directo han de tener igual longitud. Esto se consigue llenando de blancos los espacios sobrantes. Volviendo al ejemplo anterior y representando con un "_" al espacio, los registros anteriores, definidos como de longitud 11 quedarían:

"LUIS_ _ _ _ _ _ _", "JOSE_MANUEL", "PEDRO_ _ _ _ _ _"

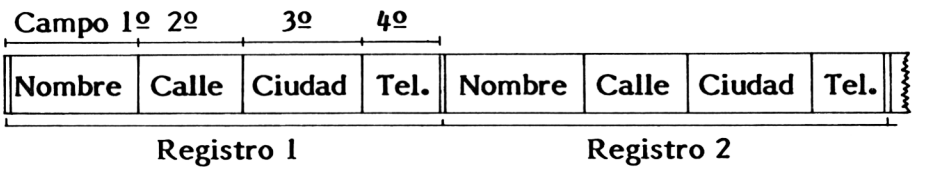
11 11 11

La necesidad de que los registros tengan la misma longitud es obvia, es la única manera de saber dónde empieza el siguiente.

Lo primero que salta a la vista es que este tipo de estructura desaprovecha espacio. Más adelante, veremos cómo se crean estos ficheros y se dimensionan sus registros, a fin de optimizar -en lo posible- el tamaño del fichero. Pero antes vamos a matizar qué es un registro y de qué está compuesto.

Un **registro** no es como en los ficheros secuenciales -un hueco donde introducir un dato- sino que contiene subapartados denominados **campos**, y en cada campo puede introducirse un dato.

La forma más sencilla de comprender esta estructura es imaginar que un registro es como una tarjeta de visita, que contiene nombre, dirección, ciudad, teléfono, etc. Pues bien, cada uno de estos apartados es un campo, y además, estos campos guardan un cierto orden establecido: primero el nombre, segundo la calle, etc. De esta forma, cada registro contendrá los siguientes campos:



Un registro puede contener perfectamente un solo campo. La situación es similar al concepto de **matriz** de una dimensión o de dos (ver cuadro a continuación).

	MATRICES	ARCHIVOS
Lista	1 dimensión	Registro con un solo campo
Tabla	2 dimensiones	Registro con varios campos

3. El programa RANDOM FILES

En el folleto que acompaña al disco que contiene el programa Random Files, vienen las instrucciones de manejo del mismo; de manera que no vamos a reproducirlas, pero sí trataremos de arrojar más luz sobre su funcionamiento, de forma que se comprenda mejor cómo trabaja, así como un programa completo que lo utiliza.

El programa Random Files está compuesto de dos partes. La primera es un programa BASIC, el **RANDOM-F.BAS**, encargado de la creación de los ficheros; y la segunda, por el programa en código máquina, **RANDOM.BIN**, encargado de la gestión de los ficheros.

Veamos ambos de forma separada.

RANDOM-F.BAS

Dado que su misión es la creación de ficheros y el consiguiente formateado, no es necesario que este programa esté contenido en el disco donde se va a crear y manipular el fichero.

Para cargarlo se utiliza, como cualquier otro programa BASIC, la orden RUN:

RUN "RANDOM-F [ENTER]

En la pantalla aparecerá el siguiente menú a rellenar con los datos apropiados:

Nombre del fichero:
Número de fichas:
Longitud de la ficha:
Disco (A/B):

Saque previamente el disco que contiene Random-F e inserte el disco de trabajo que se formateó con anterioridad, con formato de sistema.

El nombre del fichero puede ser cualquiera, respetando las reglas de sintaxis (8 caracteres máx.), pero no debe llevar indicación de tipo (nada después del punto).

El número de fichas es, en realidad, el de registros. Dicho número debe establecerse con la necesaria previsión, a fin de que no se quede pequeño ni excesivamente grande (ocupación de memoria excesiva).

La longitud de la ficha es un término algo ambiguo. Se refiere, sin duda, a la longitud del registro; pero sin embargo, no aparece por ningún lado el concepto de **campo** -que anteriormente explicamos- y es que dicho concepto se halla implícito en los ficheros creados por Random Files.

La longitud de un registro es como máximo de 255 caracteres; número que en su Amstrad aparece repetidas veces como límite y es que los ficheros creados son del tipo alfanumérico y la longitud máxima de una cadena es, precisamente, 255 caracteres. Así que, si volvemos al ejemplo de la tarjeta de visita, y suponiendo que le demos a cada registro el máximo, la distribución de los campos y su orden la establecemos nosotros, pero no el ordenador. Es decir, de cabeza o sobre el papel, pero para el ordenador no existe el concepto de campo. Esto queda para los ficheros indexados que no tienen los modelos de la serie CPC de Amstrad, aunque sí los PCW.

No obstante, se puede emular el concepto de campo de forma manual, es decir, imaginando que un registro contiene por ejemplo:

Carácter

1	100	150	200	220	250
Nombre	**Calle**	**Ciudad**	**Distrito Postal**		**Teléfono**

Contenido

El nombre abarcaría desde la posición 1 de la cadena, hasta la 99 inclusive; la calle de la 100 a la 149 y así sucesivamente. Esta organización se hará de acuerdo con el criterio que estableceremos en función de la longitud máxima de cada campo, con el fin de reducir al máximo la longitud total del registro.

El problema surge a la hora de almacenar y leer los datos. Como para el ordenador todo el registro es una palabra, debemos cuidar que la calle comience en el carácter 100; pero si el nombre sólo ocupa en un determinado registro 40 caracteres, nos sobran 59 que debemos rellenar de blancos, y así sucesivamente. De forma que cada campo debe contener tantos blancos como su longitud, menos el número de caracteres introducidos.

Veamos un ejemplo sobre la base del modelo de registro de la figura anterior.

Ejemplo de Registro

Variable Asignada	Contenido	Longitud campo	Caract. ocupad.	Blancos
N$	Antonio Fernández Gil	99	21	78
C$	Río Ebro	50	8	42
P$	Valencia	50	8	42
DP$	28.014	20	6	14
T$	2.41.41.41	31	10	21
		250	53	197
			250	

Salta a la vista que el registro está sobredimensionado, pero nos será útil a efectos de estudio (el número de blancos es casi 4 veces mayor que el de caracteres).

Tanto para añadir blancos como para leer los campos por separado, nos valdremos de las funciones de que dispone el BASIC para trabajar con cadenas.

En el ejemplo, la cadena final -tal como debe figurar en el registro- sería la agregada de las siguientes subcadenas:

N$ + "78 blancos" + C$ + "42 b." + P$ + "42 b." + DP$ +
+ "14 b" + T$ + "21 b.".

Como las longitudes de las variables N$, C$, etc. son variables para cada registro, utilizaremos la función LEN para calcular el número de blancos a añadir. En general, para un campo cualquiera representado por C$, el número de blancos necesarios para completarlo será:

Nº de blancos = Longitud del campo - LEN (C$)

En el ejemplo anterior para N$ sería:

Nº blancos de N$ = 99 - LEN (N$) = 99 - 21 = 78

como ya vimos en la tabla.

Falta, pues, resolver un único problema: cómo añadir los blancos. Para ello podemos utilizar la instrucción STRING$ o SPACE$ en las siguientes formas:

Z$ = STRING$(100," ")
Z$ = SPACE$(100)

La variable Z$ contendrá 100 espacios (número máximo previsto).

Ahora si hacemos, siguiendo el ejemplo:

N$ = N$+RIGHT$(Z$,99-LEN(A$))

obtendremos una cadena de 99 caracteres, de los cuales los 21 primeros son el nombre y el resto, espacios en blanco.

Una forma más sencilla de obtener el mismo efecto (relleno con blancos), es utilizar directamente SPACE$, sin utilizar siquiera Z$. La expresión anterior es equivalente a:

N$ = N$ + SPACE$(99-LEN(N$))

lo cual nos lleva a la expresión general de relleno:

N$ = N$ + SPACE$ (longitud campo - LEN (N$))

tomando la variable N$ en sentido genérico (cualquier campo).

Pruebe estos ejemplos y verá cómo los resultados son idénticos.

```
10 z$=SPACE$(100)
20 A$="PEPE"
30 A$=A$+RIGHT$(Z$,99-LEN(A$))
40 PRINT LEN(A$)
```

```
10 A$="PEPE"
20 A$=A$+SPACE$(99-LEN(A$))
30 PRINT LEN(A$)
```

Si procedemos de igual forma con el resto de las variables del ejemplo, podemos llegar a construir la cadena final, como agregada de subcadenas completadas con blancos:

X$ = N$ + C$ + P$ + DP$ + T$

cuya longitud sea los 250 caracteres establecidos en el ejemplo. Esta variable puede ser archivada ya en el registro, puesto que contiene los datos de los campos en el orden preestablecido, para cuya lectura e interpretación sólo necesitaremos fragmentarla nuevamente utilizando la función MID$:

MID$(X$,1,99)	Nombre
MID$(X$,100,149)	Dirección
MID$(X$,150,199)	Ciudad
MID$(X$,200,219)	Distrito postal
MID$(X$,220,250)	Teléfono

Esta fragmentación producirá subcadenas con muchos blancos, pero al igual que antes se añadieron blancos, ahora pueden quitarse. Además, el ejemplo es muy exagerado y por supuesto sus registros estarán dimensionados de forma más aprovechada.

Tras la lectura de este apartado, el lector quizá haya sacado la impresión de que el uso de estos ficheros es difícil -y en parte tiene razón si se compara con los secuenciales- pero un poco de práctica con ellos le llevará a manejarlos sin problemas.

RANDOM.BIN

En los ficheros de acceso secuencial disponíamos de comandos propios dentro del BASIC que nos permitían utilizarlos: OPENIN, WRITE, etc., y precisamente **Random.Bin** es el encargado de crear cuatro nuevas instrucciones RSX (comandos extendidos), que pueden ser utilizadas desde el BASIC anteponiéndoles el símbolo ¦ .

Para cargar estos comandos, el programa BASIC debe comenzar con las siguientes líneas:

```
10 MEMORY &9BFF
20 LOAD"RANDOM",&9C00
30 CALL &9C00
```

Con lo cual, dichas nuevas instrucciones estarán disponibles.

El programa Random.Bin debe estar en la misma cara del disco de trabajo para mayor comodidad. Utilice PIP (CP/M) para sacar una copia del disco maestro.

Esta rutina en código máquina ocupa 2.300 bytes, incluida la tabla de información de cada uno de los 15 ficheros que pueden estar abiertos de forma simultánea (en los secuenciales sólo dos: uno de lectura y otro de escritura), y también incluye un buffer de 512 bytes para almacenar datos transitorios, con el fin de minimizar el número de accesos al disco.

Dicha tabla comienza en la dirección &H9D1D en los 664 y 6128, y en la &H9D0C para el 464. Asimismo, el citado buffer comienza en la &HA2F1 para 664 y 6128; y &HA2D4 para el 464.

Para una mayor información, incluimos a continuación la tabla de datos mencionada, en la cual se indican las posiciones de memoria según el modelo de ordenador.

TABLA DE INFORMACION SOBRE LOS FICHEROS

M O D E L O 664 Y 6128	464	C O N T E N I D O
&H9D1D	&H9D0C	LONGITUD DE LOS REGISTROS DEL FICHERO No.1
&H9D1E	&H9D0D	INDICATIVO DE LA UNIDAD (0="A", 1="B").
&H9D1F	&H9D0F	PISTA DE UBICACION DEL FICHERO No.1
&H9D20	&H9D0F	SECTOR DE UBICACION DEL FICHERO No.1
....	....	.....
&H9D55	&H9D44	LONGITUD DE LOS REGISTROS DEL FICHERO No.15
&H9D56	&H9D45	INDICATIVO DE LA UNIDAD
&H9D57	&H9D46	PISTA DE UBICACION DEL FICHERO No.15
&H9D58	&H9D47	SECTOR DE UBICACION DEL FICHERO No.15

* Abrir un fichero

En los ficheros de acceso directo sólo existe una orden OPEN, cuya sintaxis es la siguiente:

¦ OPEN,@N$,NF,LR,ND

donde **N$** es el nombre del fichero (sin tipo)
NF número del archivo (1 a 15)
LR longitud de los registros (igual a la que se indicó cuando se creó el archivo en RANDOM.BAS)

ND unidad de disco (1 para A y 2 para B) en la que se va a acceder al fichero.

* Cerrar un fichero

Sólo existe una orden con dos formatos y de sintaxis mucho más sencilla que la anterior:

¦CLOSE

que cierra todos los ficheros abiertos y

¦CLOSE NF

que cierra sólo el archivo número NF de los 15 posibles.

* Leer un registro

Para leer un registro se utiliza el comando extendido ¦READ, con la siguiente sintaxis:

¦READ,@X$,NR,NF

donde **X$** es la variable en la cual se va a depositar el contenido del registro
NR es el número del registro que se quiere leer
NF número del fichero (uno de los 15 posibles).

Dado que en X$ se encuentran implícitos todos los campos, se puede obtener el contenido de cada uno de ellos mediante la función de fragmentación MID$ (tal como se indicó anteriormente). En el apéndice puede ver cómo utilizarla con estos fines.

* Grabar un registro

Se utiliza el comando extendido ¦ WRITE con la siguiente sintaxis:

¦WRITE,@X$,NR,NF

Como ve, los parámetros son los mismos que los utilizados para **leer.**

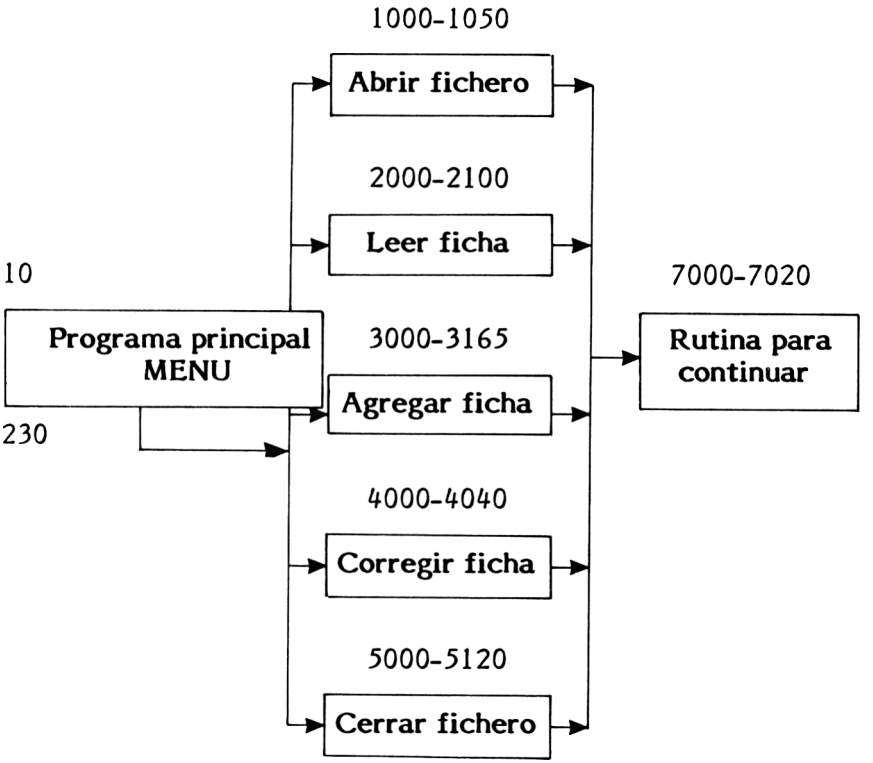
4. Apéndice al Capítulo

Preliminares

Con el fin de ver -de forma práctica- el conjunto de conceptos estudiados en el capítulo, vamos a desarrollar un programa de Agenda de direcciones.

Se ha creído oportuno incluir como ejemplo de ficheros de acceso directo, un programa igual al desarrollado en el capítulo de los ficheros secuenciales, manteniendo la misma estructura general; lo cual permitirá, sin duda, un estudio comparativo más provechoso.

La organización general del programa responde básicamente al siguiente organigrama:



Los flujos indicados son los principales, de forma que los retornos se realizan en sentido contrario.

* Bases de partida

Antes de elaborar un programa para el manejo de ficheros de acceso directo, es necesario definir sobre el papel las premisas de dicho fichero: número de fichas, longitud de los registros, campos, etc.

En el ejemplo que proponemos, se han utilizado los siguientes datos de partida.

- El número máximo de fichas será de 50.
- Cada ficha contendrá los siguientes campos con la longitud que se indica.

	Longitud	Comienzo/fin
* Nombre y apellidos	30	1 - 29
* Calle, número, piso	40	30 - 69
* Población	15	70 - 84
* Provincia	15	85 - 99
* Distrito postal	5	100 - 104
* Teléfono y pref. o ext.	15	105 - 120
TOTAL	120	

- La longitud de los registros será pues de 120 caracteres, repartidos en campos, según se indica en la tabla.

* Trabajos previos

1 Formatear una cara de un disco virgen o que no necesite su contenido, utilizando formato de sistema (DISCKIT2, DISCKIT3 o FORMAT de CP/M).

2 Copie el programa RANDOM-F.BAS y RANDOM.BIN en la cara formateada. Para ello puede utilizar el programa PIP de CP/M en la siguiente forma:

Introduzca la cara 1 de los discos de sistema; entre en CP/M y teclee tras la impronta la orden:

```
A > PIP
```

Una vez aparezca el indicativo * introduzca la orden

```
B:=A: RANDOM-F.BAS
```

Introduzca el disco que contiene RANDOM-F y presione ENTER. Siga las instrucciones de cambio de disco, teniendo en cuenta que **A:** será el disco original que contiene RANDOM-F; y **B:** será el disco que se formateó y sobre el que se realizará la copia.

Una vez hecha la copia, aparecerá de nuevo la impronta *. Repita la operación para RANDOM.BIN.

Con CP/M 2.2 puede utilizar FILECOPY (CPC664 y 464 con unidad de disco).

3 Haga RUN"RANDOM-F para crear el fichero; conteste a las cuestiones planteadas introduciendo los datos obtenidos en "Bases de partida":

Nombre del fichero:	utilice por ejemplo "AGENDA"
Número de fichas:	50 o más si las necesita (50=6K)
Longitud de ficha:	120
Disco (A/B):	A si sólo dispone de una unidad de disco

De esta forma el disco contendrá:

RANDOM.BIN
RANDOM-F.BAS
AGENDA

Para verificarlo haga un CAT o DIR (CP/M).

Ahora siga atentamente las explicaciones del programa que se describe y una vez grabado, estará listo para su uso.

Nota: En realidad, el programa RANDOM-F no es estrictamente necesario copiarlo, pero le resultará cómodo tenerlo en el mismo disco.

Descripción del programa

* Programa principal. MENU

Tal y como se indicó anteriormente, el programa comienza con la carga de RANDOM.BIN (líneas 22 a 24). Se ha incluido la línea 21 con el fin de que en las sucesivas pruebas (RUN sucesivos), no se cargue dicha rutina repetidas veces.

El valor de la posición de memoria &9C00 será 1 si RANDOM.BIN ya fue cargado, por lo que en este caso, se transfiere el control a la línea 120 (comienzo del Menú).

Si por diversas razones necesita parar el programa, provocando un BREAK, arranque siempre con GOTO 120. Haciéndolo así podrá eliminarse la línea 21, aunque es preferible dejarla -por si acaso.

El menú y direccionamiento a las rutinas es idéntico al utilizado en el programa AGENDA del Apéndice al capítulo 4 -Ficheros secuenciales- aunque se han recortado opciones.

*** Abrir fichero**

Esta rutina es la encargada de abrir el fichero creado previamente por RANDOM-F.

El programa solicita el nombre del fichero, que debe ser el mismo que se utilizó en su creación.

El IF de la línea 1030 comprueba si el fichero ya estaba abierto. La dirección de memoria indicada, corresponde a los CPC664 y 6128; en caso de utilizar un 464 cámbiela por:

664/6128	464
&H9D1D	&H9D0C

La línea 1040 es la encargada de la apertura. F$ contiene el nombre del fichero. El número del archivo es 1 (sólo hay uno). La longitud de los registros es 120 y la unidad de discos es la A (valor 1).

*** Leer ficha**

Para leer una ficha (registro), es necesario conocer el orden que ocupa. Por ello, la 2010 solicita dicho número de orden.

La 2020 se encarga de la lectura, en X$ se almacena el contenido de la ficha N del fichero 1.

Con el fin de presentar los datos en forma de ficha, en vez de todo seguido (tal y como figura en el archivo), procedemos a fragmentar la cadena X$ en la forma prevista en las "Bases de partida".

La línea 2100 contiene las parejas de valores que determinan cada campo. El primer número indica el comienzo del campo y el segundo su longitud.

Por ejemplo, el primer campo -nombre y apellidos- viene identificado por la pareja 1,29 (1 posición de comienzo, 29 longitud), de tal forma que MID$(X$,1,29), nos dará la subcadena que contiene el nombre y apellidos.

* Agregar ficha

Esta opción permite introducir nuevas fichas en el fichero. En primer lugar se produce la entrada de datos. Cada dato se introduce mediante 3 líneas de programa, por ejemplo:

3015 Escribe el título del campo.
3016 Imprime una cadena de subrayado igual a la longitud del campo, a fin de guiarnos y no sobrepasarla.
3017 Sube el cursor arriba para que los datos del INPUT se escriban sobre los subrayados.

La línea 3070 controla que ningún campo haya sobrepasado la longitud prevista.

De la 3080 a la 3140 se procede a rellenar con blancos hasta la longitud prevista en cada uno de los campos.

La 3140 efectúa el agregado de campos para formar el registro, tal y como se introducirá en el fichero.

La 3150 comprueba que no se han sobrepasado los 120 caracteres. Si el programa está bien, no debe producirse este tipo de error. No obstante, debe incluirse a fin de filtrar en lo posible todos los errores, cualquiera que sea su naturaleza u origen.

Por último, la 3160 se encarga de realizar la grabación con la misma sintaxis que la lectura.

* Corregir una ficha

Esta rutina se apoya en dos anteriores, puesto que para corregir una ficha, lo primero es ver cómo está. Se llama a la 2000 "Leer datos" y después, puesto que existe otra rutina de "Entrada de datos", se la llama entrando por la línea 3015, dado que el número de ficha es conocido.

* **Cerrar el fichero**

Se utiliza la forma más general, cerrar todos.

```
¦ CLOSE
```

Todas estas rutinas terminan con una llamada a la 7000, que emite un mensaje en ventana y espera a que se pulse una tecla. En todos los casos se retorna al menú para seguir trabajando.

A continuación se expone el listado de todo el programa, con las rutinas debidamente ensambladas. Esperamos que su estudio le permita crear otros ficheros de bibliotecas, alumnos, etc.

Como podrá ver, estos ficheros son mucho más sencillos de manejar de lo que en principio parece y, desde luego, mucho menos rígidos que los secuenciales.

A fin de hacer el ejemplo lo más simple posible, no se han incluido rutinas de ordenación ni de salida por impresora, etc. Dejamos esto como trabajo del lector, apoyándose en los diversos temas tratados a lo largo del libro.

Notas: El listado que se acompaña, contiene varias sentencias PEEK que corresponden a los modelos CPC664 y 6128; para su utilización por el CPC464 deben variarse las direcciones que se indican a continuación:

Línea	CPC464	CPC664 y 6128
1030	&9D0C	&9D1D

```
5 ´ ***********************************************
10 ´ ****  AGENDA DE DIRECCIONES Y TELEFONOS   ****
15 ´ ***************   AGENDIR   ******************
20 ´ *****  Archivo de acceso directo (disco) *****
21 IF PEEK (&9C00)=1 THEN 120
22 MEMORY &9BFF
23 LOAD "RANDOM",&9C00
24 CALL &9C00
40 MODE 1
50 ´Pantalla para mensajes
60 WINDOW #1,1,40,24,25
70 PEN #1,0:PAPER #1,1
120 CLS:´ ###########   M E N U     ################
130 LOCATE 12,3: PRINT "M E N U"
135 LOCATE 6,5: PRINT CHR$(24);"1. ABRIR FICHERO";CHR$(24)
140 LOCATE 6,7: PRINT "2. LEER FICHA"
160 LOCATE 6,9: PRINT "3. AGREGAR FICHA"
180 LOCATE 6,11: PRINT "4. CORREGIR FICHA"
190 LOCATE 6,13: PRINT CHR$(24);"5. CERRAR FICHERO";CHR$(24)
200 INPUT #1," SELECCIONE OPCION ";OP
210 CLS#1:IF OP<1 OR OP>5 THEN PRINT CHR$(7):GOTO 200
220 ON OP GOSUB 1000,2000,3000,4000,5000
230 GOTO 120
1000 ´ ABRIR EL FICHERO ***************************
1010 CLS:CLS#1:INPUT #1,"NOMBRE DEL FICHERO";F$
1020 IF LEN(F$)>8 THEN 1010
1030 IF PEEK (&9D1D)<>0 THEN PRINT #1,"EL FICHERO YA ESTA AB
IERTO":GOTO 1050
1040 |OPEN,@F$,1,120,1
1050 GOSUB 7000:RETURN
2000 ´ LEER FICHA  ********************************
2010 CLS:INPUT "FICHA NUMERO ";N
2020 |READ,@X$,N,1
2030 RESTORE 2100
2040 FOR K=1 TO 6
2050 READ P,F
2060 PRINT MID$(X$,P,F)
2070 NEXT K
2080 GOSUB 7000:RETURN
2100 DATA 1,29,30,40,70,15,85,15,100,5,105,15
3000 ´ AGREGAR FICHA  ****************************
3010 CLS:INPUT "FICHA NUMERO";N
3015 PRINT "Nombre y apellidos"
3016 PRINT STRING$(29,"_")
3017 PRINT CHR$(11);:INPUT "",N$
3020 PRINT "Calle numero y piso"
3022 PRINT STRING$(40,"_")
3025 PRINT CHR$(11);CHR$(11);:INPUT "",C$
```

```
3030 PRINT "Poblacion"
3032 PRINT STRING$(15,"_")
3035 PRINT CHR$(11);:INPUT "",p$
3040 PRINT "Provincia"
3042 PRINT STRING$(15,"_")
3045 PRINT CHR$(11);:INPUT "",PR$
3050 PRINT "Distrito Postal"
3052 PRINT STRING$(5,"_")
3055 PRINT CHR$(11);:INPUT "",DP$
3060 PRINT "Telefono y prefijo o extension"
3062 PRINT STRING$(15,"_")
3065 PRINT CHR$(11);:INPUT "",T$
3070 IF LEN(N$)>29 OR LEN(C$)>40 OR LEN(P$)>15 OR LEN(PR$)>1
5 OR LEN(DP$)>5 OR LEN(T$)>15 THEN PRINT #1,"TEXTOS INCORREC
TOS":GOTO 3015
3080 N$=N$+SPACE$(29-LEN(N$))
3090 C$=C$+SPACE$(40-LEN(C$))
3100 P$=P$+SPACE$(15-LEN(P$))
3110 PR$=PR$+SPACE$(15-LEN(PR$))
3120 DP$=DP$+SPACE$(5-LEN(DP$))
3130 T$=T$+SPACE$(15-LEN(T$))
3140 X$=N$+C$+P$+PR$+DP$+T$
3150 IF LEN(X$)>120 THEN PRINT #1,"EL REGISTRO TIENE MAS DE
120 CARACTERES. REVISAR PROGRAMA"
3160 |WRITE,@X$,N,1
3165 PRINT #1,"FICHA No.";N;" GRABADA EN FICHERO"
3170 GOSUB 7000:GOTO 120
4000 ' CORREGIR UNA FICHA  ************************
4010 GOSUB 2000
4020 PRINT "INTRODUZCA   N U E V O S   D A T O S"
4030 GOSUB 3015
4040 GOSUB 7000:RETURN
5000 ' CERRAR FICHERO  ****************************
5010 |CLOSE
5020 PRINT #1,"FICHERO CERRADO"
5120 GOSUB 7000:RETURN
7000 ' RUTINA DE CONTINUACION  ********************
7010 PRINT #1,"PRESIONE UNA TECLA PARA CONTINUAR":CALL &BB18
7020 RETURN
```

CAPITULO 6

FICHEROS CREADOS POR EAMSWORD

1. Generalidades
2. Uso de los Ficheros
3. Ejemplo

Ficheros creados por EAMSWORD

1. Generalidades

El programa **EAMSWORD** es un sencillo tratamiento de textos (word processing o text processing) incluido en el mismo disco que RANDOM FILES y que se acompaña en la compra de los equipos.

También existe una versión en cinta, para el CPC464, válida incluso para los modelos de disco.

No vamos a explicar, ni mucho menos, cómo se utiliza, puesto que ello escapa del principal objetivo de este libro que es el estudio de ficheros. Además, lo sencillo de su manejo hace que sean suficientes las explicaciones contenidas en el folleto adjunto al programa.

Nuestro interés por el programa no es otro que el de tratar de hacer un uso más amplio de los ficheros creados por él.

Cuando introducimos un texto mediante el programa EAMSWORD y lo grabamos en cinta o disco, obtenemos un fichero de tipo secuencial, de manera que con un poco de habilidad -en principio- parece posible utilizarlo como cualquier fichero secuencial creado desde BASIC.

Vamos a estudiar cómo son estos ficheros y, sobre todo, cómo se almacena la información. Para ello no vamos a realizar una disertación teórica sobre el mismo, sino que, a través de un ejemplo, vamos a sacar nuestras propias conclusiones y de ellos su posible utilidad fuera del EAMSWORD.

Carguemos el programa utilizando RUN "EAMSWORD". Una vez se nos ofrezca el MENU, presionaremos **C** para crear un documento al que llamaremos **prueba**. Trabajemos en 80 columnas.

Introduzca después el siguiente texto, en la forma que se indica.

En ↵
un ↵
lugar ↵ ↵ = ENTER o RETURN
de ↵
la ↵
Mancha ↵

Tras presionar el último ENTER (después de la palabra "Mancha"), salga del editor (CONTROL↑), seleccione **S** para salvar y después **P** para proceder a la grabación.

Después de estas operaciones tenemos un fichero, de nombre **prueba** grabado con el tratamiento de textos. Ahora bien, ¿cómo es el fichero? ¿Es un fichero secuencial de forma que las palabras se encuentran en la siguiente relación?

En	**un**	**lugar**	**de**	**la**	**Mancha**

Estas preguntas no son difíciles de contestar. Para ello vamos a valernos de la orden TYPE de CP/M, que produce el volcado de un fichero en la pantalla o impresora, como ya se indicó anteriormente.

Para seguir nuestro ejemplo, introduzca el disco de sistema, resetee el ordenador y llame a CP/M. Cuando aparezca la impronta saque el disco CP/M, introduzca el disco en que grabó **prueba** y teclee la orden

TYPE PRUEBA

y obtendrá en pantalla (e impresora, si previamente presionó CONTROL P) los siguientes datos:

```
A> type prueba
   "prueba",6,72,60,"Y",12,"
                                    ",5,2,"N",0
   27
   64
   0
   0
   0
   0
   0
   0
   0
   0
```

```
"En"
"un"
"lugar"
"de"
"la"
"Mancha"
""
```

Analicemos estos datos. En primer lugar viene el nombre del archivo, después una serie de números, con el siguiente significado:

6	Número de líneas de texto
72	Número de caracteres por línea
60	Número de líneas por página
Y	Justificación del margen derecho
12	Salto de página (valor ASCII)
	Espacio
5	Margen izquierdo
2	Modo de escritura (80 columnas)
N	Doble espacio
0	Primer página

Después en columna

27	Set de caracteres para la impresora
64	
0	(no utilizados en el ejemplo)
0	
0	
0	
0	
0	
0	
0	

A continuación las 6 líneas de texto y por último dos comillas solas, es decir, una cadena vacía.

Lo primero que salta a la vista es que los textos aparecen entre comillas y los números sin ellas.

Por otro lado, los datos van separados por comas o en líneas independientes. ¿Sería posible utilizar este fichero desde BASIC? De este tema trataremos en el siguiente apartado.

2. Uso de los ficheros

Nos hemos propuesto anteriormente dar una utilidad adicional a estos ficheros y nos hemos planteado su utilización desde el BASIC. Hagamos una prueba con el ejemplo de que disponemos y verifiquemos esta posibilidad.

Mediante este programa abriremos el fichero para leer e imprimir en pantalla todos sus datos, en el orden en que están escritos:

```
10 ´ LECTURA FICHERO EAMSWORD
20 ´ ........... EA1 .........
30 OPENIN "PRUEBA"
40 WHILE NOT EOF
50 K=K+1
60 INPUT #9,A$
70 PRINT "DATO No.";K;" --- ";A$
80 WEND
90 CLOSEIN
```

Una vez tecleado, haga RUN y obtendrá en pantalla la siguiente lista de datos ordenada.

```
  Dato 1    prueba
Dato 2    6
Dato 3    72
Dato 4    60
Dato 5    Y
Dato 6    12
Dato 7

Dato 8    5
Dato 9    2
Dato 10    N
Dato 11    O
Dato 12    27
Dato 13    64
Dato 14    0
Dato 15    0
Dato 16    0
Dato 17    0
Dato 18    0
Dato 19    0
Dato 20    0
Dato 21    0
```

```
Dato 22    En
Dato 23    un
Dato 24    lugar
Dato 25    de
Dato 26    la
Dato 27    Mancha
Dato 28
```

Las primeras conclusiones son obvias.

1ª El fichero es manejable desde BASIC -que es uno de nuestros objetivos.

2ª Se compone (en este caso) de 28 líneas o registros.

3ª El texto comienza en el dato número 22 y acaba 6 después, (en el 27), puesto que 6 es su longitud en líneas. Por tanto, si consideramos una línea como registro, nuestro fichero contiene 6 registros.

4ª El último dato es un registro vacío, ajeno a nuestro fichero **(fin de fichero).**

Pero hay muchas más consecuencias que se desprenden de este hecho, puesto que en muchas ocasiones resulta más sencillo crear un fichero como si fuese un texto, que a través de una entrada de datos de un programa BASIC. La corrección de datos del fichero es también muy cómoda, a través al modo de inserción del editor, y del movimiento mediante los cursores.

Otra ventaja adicional es que el número de datos que contiene el fichero (datos propiamente dichos) viene incluido en el propio fichero (dato 2º).

Hay también datos que resultan superfluos, como el número de caracteres por página, set de códigos, de la impresora, etc., y que por tanto deben ser desechados en nuestra aplicación.

Veamos una rutina de lectura de un fichero BASIC más elaborada que la anterior, que almacena los datos útiles del fichero en una matriz, para su posterior manipulación.

```
10 ' LECTURA FICHERO EAMSWORD
20 ' ........... EA2 .........
30 INPUT "NOMBRE DEL FICHERO "F$
40 OPENIN F$
50 INPUT #9,F$,N
60 DIM A$(N)
70 FOR K=1 TO 19:INPUT #9,A$(0):NEXT
80 FOR K=1 TO N
90 INPUT #9 A$(K)
100 NEXT K
110 CLOSEIN
```

La línea 50 procede a la lectura del nombre del fichero (F$) y del número de líneas (datos) del mismo, almacenando este dato en N.

Dado que de los 21 primeros -menos significativos- se han leído los dos primeros. La línea 70 establece un bucle de lectura de 19 vueltas, que almacena la información en el elemento cero de la matriz (A$(0)).

Una vez "saneado" el fichero comienza la lectura de datos "útiles", para lo cual se establece el bucle de la línea 80, dado que el límite es ya conocido.

Bien, hemos visto cómo podemos utilizar desde BASIC, la información de un fichero creado por EAMSWORD. Profundicemos un poco más a fin de tratar de sacar de esta posibilidad el máximo provecho.

En principio, y dado el ejemplo que hemos utilizado, pudiera parecer que en estos ficheros únicamente pueden almacenarse listas de datos, pero no tablas o lo que, hablando en términos de ficheros, serían respectivamente registros con un solo campo o registros con varios campos.

Si leyó previamente el capítulo dedicado a los **Ficheros de Acceso Directo,** tendrá ya una idea de cómo trabajar con varios campos y recordará, sin duda, la tarea de rellenar con blancos cada campo, a fin de tomar la longitud prevista.

Piense ahora en términos de tratamiento de textos, qué ocurre cuando utilizamos el tabulador. El cursor se desplaza en un punto fijo dejando "blancos", entre lo escrito anteriormente y el punto marcado de tabulación. ¿No cree que existe una analogía con la forma en que tratábamos los registros con campos?

3. Ejemplo

Vamos a utilizar un ejemplo y veamos las posibilidades de la utilizacion del tabulador como limitador de campo.

Supongamos la siguiente tabla de datos:

NOMBRE	EDAD	ESTATURA	PESO
Antonio	14	1,60	55
José Luis	13	1,57	52
Pedro	9	1,58	45

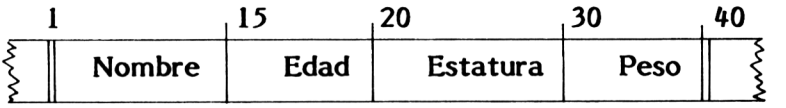
Tenemos 3 filas x 4 columnas pensando en términos matriciales. Como fichero, cabe suponer que existen 3 registros y en cada registro 3 campos.

Si pensáramos utilizarlo como fichero de acceso directo, lo primero sería calcular las longitudes de los campos para conocer la longitud de los registros.

El número máximo de caracteres de cada campo es:

Nombre	9
Edad	2
Estatura	4
Peso	2

De forma que podemos construir un registro, que esté formado de la siguiente forma (por exceso):



La longitud que se ha asignado a cada campo es muy superior a la estricta pero sirve a los fines del ejemplo y, por otro lado, no resulta aconsejable ser muy preciso en el dimensionamiento, pensando que un fichero grande siempre contiene datos más o menos anómalos, en el sentido de que se apartan bastante de las dimensiones medias aunque, repetimos, no es el caso del ejemplo.

Volvamos al tratamiento de textos. Creemos el fichero **Prueba 2** e introduzcamos los datos con ayuda del tabulador, es decir, presionemos **control 0** en las posiciones 15, 20 y 30 (trabajaremos en MODO 2).

Al teclar los datos con ayuda de TAB, comprobaremos que este tipo de entrada de datos resulta muy cómodo, todo está a la vista y ordenado, no como en una entrada normal, donde sólo vemos la ficha que estamos introduciendo y además no puede corregirse con la misma facilidad que con el editor de tratamiento de textos.

Grabe el fichero en el disco y, utilizando el programa anterior, podrá obtener los datos desde BASIC. Ahora bien, imaginemos que lo que deseamos no es listar el fichero -cosa que ya conocemos- sino realizar cálculos con parte de sus datos. Por ejemplo:

> Calcular la media de las edades y de los pesos, e indicar el nombre y estatura del alumno más alto.

Basándonos en los programas anteriores para la lectura de archivo y en los del capítulo anterior en cuanto a fragmentación, se ha elaborado el siguiente programa:

```
10 ´ CALCULOS CON FICHEROS EAMSWORD
20 ´ ........... EA3 ...............
30 ´ Lectura  de datos del fichero
40 INPUT "NOMBRE DEL FICHERO ";F$
50 CLS
60 OPENIN F$
70 INPUT #9,F$,N
80 PRINT "Fichero *** ";f$;" ***"
90 PRINT "No. de lineas=";n
100 PRINT "-----------------------------"
110 DIM A$(N)
120 FOR K=1 TO 19:INPUT #9,A$(0):NEXT
130 FOR K=1 TO N
140 INPUT #9, A$(K)
150 NEXT K
160 CLOSEIN
170 ´ Calculo de las medias
180 FOR K=1 TO N
190 ME=ME+VAL(MID$(A$(K),15,4))
200 MP=MP+VAL(MID$(A$(K),30,10))
210 NEXT K
220 PRINT "MEDIA DE EDADES=";ME/N
```

```
230 PRINT "MEDIA DE PESOS =";MP/N
240   Mayor de estatura
250 FOR K=1 TO N-1
260 IF MID$(A$(K),20,9)>MID$(A$(MAY),20,9) THEN MAY=k
270 NEXT K
280 PRINT "EL MAYOR ES ";MID$(A$(MAY),1,14);
290 PRINT " QUE MIDE ";MID$(A$(MAY),20,9)
```

Observe cómo el cálculo de las medias se realiza acumulando en una variable la suma de todas las edades y de todas los pesos (ME, MP), obteniendo los sumandos por partición de A$(K) entre los límites 15 a 19 y 30 a 39, respectivamente, dado que dichas subcadenas contienen los datos seguidos de blancos, que son transformados a valor numérico mediante la función VAL.

Lo expuesto es sólo una pequeña muestra de las enormes posibilidades de los ficheros creados por EAMSWORD que, si bien como tratamiento de textos está superado por otras versiones, nos permite trabajar cómodamente en nuestros fines.

CAPITULO 7

FICHEROS CREADOS POR CP/M

1. **Introducción**

2. **El Editor de CP/M**

   * Generalidades
   * Manejo del editor. Operaciones básicas
   * Otras operaciones
   * Resumen de órdenes del editor ED.COM

3. **Utilización de los Ficheros ED desde BASIC**

Ficheros creados por CP/M

1. Introducción

En el capítulo anterior hemos visto cómo un fichero creado por un programa de tratamiento de textos, es utilizable desde BASIC.

El sistema operativo CP/M con que cuenta su Amstrad, dispone, entre otros, de un programa editor de textos. No es propiamente un tratamiento de textos, pero sí permite introducir de forma análoga a éste, una serie de datos (textos) que serán posteriormente almacenados en un fichero. Su operatividad y funciones son muy distintas, pero bajo nuestro punto de vista -de utilización de ficheros- puede sernos igualmente útil.

2. El Editor de CP/M

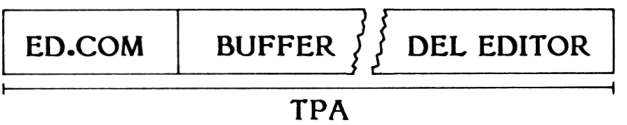
Generalidades

Antes de estudiar con detenimiento los ficheros creados por el editor de CP/M, pasemos a una somera descripción de en qué consiste el editor CP/M y cómo se utiliza -dado que no se incluye en el manual.

El programa editor es el transitorio ED.COM. Para utilizarlo debe primero entrar en CP/M y cuando obtenga la impronta A > teclee el nombre seguido de ENTER

A > ED.COM

El programa se instala en la base del TPA de la memoria RAM, de forma que el resto constituye el buffer del editor.



Un fichero de textos se compone de caracteres y éstos se agrupan en líneas -como vimos en el programa EDMSWORD- igualmente ocurre con los ficheros creados por ED de CP/M.

La separación de líneas se produce mediante dos caracteres de control: **^M** y **^J** que corresponden al retorno del carro y a la alimentación de línea, respectivamente. Ambos caracteres son generados de forma automática por ED cuando se pulsa la tecla ENTER (RETURN).

Al cargar ED observará que desaparece la impronta de CP/M, -A > - y aparece el símbolo *. Este es la impronta del editor. Su aparición indica que nos encontramos a **nivel de editor,** es decir, listos para trabajar con él.

El programa ED.COM se encuentra en la cara 1 del disco de sistema en el CPC664 y en las 1 y 4 del 6128. (En éste último la cara 1 contiene la versión CP/M Plus y la 4 la de CP/M 2.2).

Para comenzar a trabajar con ED debe previamente realizar una copia de este programa en un disco debidamente formateado. Utilice FILECOPY en CP/M 2.2 o PIP en CP/M Plus (como se indica en el cuarto apartado del capítulo 5: Trabajos previos).

Dado que en el manual no se indica cómo funciona el editor, pasamos a exponer brevemente sus principales comandos. (Para mayor información puede consultar diversas obras sobre CP/M).

Nota: El símbolo ^ indica que debe presionarse la tecla CONTROL simultáneamente a la letra que sigue.

Manejo del editor. Operaciones básicas

Son muy variadas las funciones del programa ED. Veamos a continuación las principales.

* Llamar al editor y crear un fichero

Ya hemos visto en el apartado anterior que el programa se invoca mediante:

A > ED.COM

Pero lo más útil es simultanear la llamada con la creación (apertura) del fichero que contendrá los datos. Esto se consigue añadiendo el nombre del fichero a la orden ED, en la misma línea; es decir, si creamos un fichero de nombre **PRUEBA.UNO,** utilizaremos:

A > ED PRUEBA.UNO

De esta forma, se realiza simultáneamente la carga de ED.COM y la creación del archivo **PRUEBA.UNO.**

En pantalla obtendrá como respuesta:

NEW FILE
: *

La primera línea indica que el fichero no existía: es nuevo; y la segunda es la impronta propia del editor -como se ha indicado.

*** Introducir texto. Inserción**

Para comenzar a introducir textos, tal como se haría con EAMSWORD, debe indicarse al editor que se va a **insertar** texto. Para ello, tras la impronta anterior, añadimos "i" seguido de ENTER, hágalo y obtendrá como respuesta:

: * i
1 :

indicando que está dispuesto para recoger la primera línea. Introduzcamos el mismo ejemplo que utilizó con EAMSWORD. Teclee:

1: En
2: un
3: lugar
4: de
5: la
6: Mancha
7:

Tras la introducción de la última línea, (la 6), aparecerá el indicativo **7:;** es decir, se encuentra en espera de más líneas, debemos, pues, indicarle que la entrada de datos ya ha finalizado, saliendo del **modo inserción** en que nos encontramos.

* **Salir del modo inserción**

Para abandonar el modo inserción, basta con presionar ^Z (^ = CONTROL). Tras ello volverá a reaparecer la impronta del editor:

haga 7: ^Z

y la pantalla quedará en la forma siguiente:

```
:
:
6: Mancha
7:
: *
```

* **Grabar el fichero**

Puesto que hemos terminado con la introducción de líneas, procedemos a grabar su contenido en el fichero que fue creado previamente al efecto.

La orden de grabación es:

```
: * e
```

Al pulsar ENTER verá que el disco es activado y en la pantalla volverá a reaparecer la impronta de CP/M, A >. El control ya no está bajo ED, sino bajo el sistema operativo.

Utilice la orden DIR y obtendrá:

```
ED      . COM
PRUEBA . UNO
PRUEBA . BAK
```

El programa **PRUEBA.BAK** tiene su origen en un fichero transitorio creado por la orden **ED PRUEBA.UNO,** que crea automáticamente el fichero **PRUEBA.$$$** que posteriormente pasará a consolidarse como **PRUEBA.UNO,** por la orden de grabación **"e",** por lo que **PRUEBA.$$$** pasa a ser **PRUEBA.BAK** (duplicado).

Si lo desea puede listar el contenido del fichero **PRUEBA,** mediante la orden permanente TYPE (CP/M).

A > TYPE PRUEBA.UNO

Otras operaciones

Las operaciones y funciones del editor ED.COM de CP/M son muy variadas y su estudio escapa de la finalidad práctica de este libro. Por ello, remitimos al lector a la bibliografía específica para un estudio profundo sobre este editor.

No obstante, vamos a ver a continuación algunas órdenes complementarias, que permiten la inserción y borrado -fundamentales para trabajar con ED.

* Inserción de líneas

Partiendo del fichero creado anteriormente y guardado en disco bajo el nombre **PRUEBA.UNO,** procedemos primeramente a llamar al editor indicándole el nombre del fichero en cuestión:

```
A > ED PRUEBA.UNO
: *
```

Dado que el fichero ahora sí existe, no aparece el mensaje **NEW FILE.** Pero esta orden, por sí sola, no carga el contenido del fichero en el buffer del editor, para ello hemos de añadir:

```
# a
```

Al presionar ENTER se realiza la transferencia disco-buffer ED, tras la cual aparece:

```
1: *
```

indicando que la orden ha sido ejecutada, aun cuando no se muestre el texto en pantalla. Para visualizarlo se utiliza la orden # t (tipografiar), tras la cual, en pantalla se obtiene:

```
A > ED PRUEBA.UNO
: * #a
1: *  #t
1: En
2: un
3: lugar
4: de
5: la
6: Mancha
1: *
```

La expresión **1: *** que figura en último lugar, es indicativa de que cualquier inserción sería colocada en primer lugar, es decir, antes de la línea **"En"**. Probemos:

```
1: * i
1: D. QUIJOTE
2:
```

Si salimos con ^Z volviendo al editor y grabando el nuevo archivo con **"e"**, obtendremos el fichero incrementado en una línea.

Pero en líneas generales, la inserción ha de realizarse en cualquier parte. Para ello tras la expresión **1: *** -que vimos anteriormente- debe indicarse el número de línea a insertar, seguido de dos puntos. Por ejemplo, si entre la línea 5 y 6 queremos introducir la palabra **"extensa"**, debemos proceder de la siguiente forma:

```
1: * 6:
6: extensa
7:
```

Si no deseamos insertar más, saldremos del **modo inserción** con ^Z, como ya se indicó anteriormente.

Los errores mecanográficos al escribir las líneas, pueden ser subsanados mediante **^H**, que es la orden específica de borrado de CP/M, cuyo efecto es eliminar el carácter situado a la izquierda del cursor. En el epígrafe siguiente se describen con más detalle las posibilidades de borrado y el movimiento de los cursores.

* Borrado parcial de líneas. Movimiento de los cursores

Para poder actuar sobre una determinada línea, debemos previamente, situar el puntero en dicha línea, valiéndonos para ello de la orden **n:**, siendo **n** el número de línea en cuestión.

Una vez situados, ED dispone de una orden específica de borrado parcial de líneas o conjunto de caracteres. Su sintaxis es la siguiente:

: ± n d

Donde **n** indica el número de caracteres a suprimir, y la letra **d** es la inicial de **delete** (borrar). El signo + o - indica la dirección en que se va a realizar el borrado; (-) a la izquierda del cursor, (+) a la derecha. Si se omite el signo, se toma (+) por defecto.

Ahora bien, además de la orden **n:** para el desplazamiento de líneas, ED dispone de un juego de órdenes que permiten desplazamientos de los cursores en los cuatro sentidos: arriba-abajo (líneas), derecha-izquierda (en una línea). Veamos estas órdenes:

: ± b

b es la inicial de **begin** o **bottom** (comienzo o fondo). La función asumida es situar el puntero al principio o al final del fichero, según que el signo sea - o +, respectivamente. Por omisión se toma el signo +:

: ± n l

l es la inicial de **line** (línea). Esta orden permite avanzar o retroceder un número de líneas determinado por el número **n**, según el signo empleado, como en la orden anterior.

El número **n** puede ser sustituido por el comodín #, que equivale a "todo", de manera que: #l hace avanzar al puntero hasta el final y : - #l, al principio:

: ± n c

c es la inicial de **character** y la función asumida es mover el cursor a lo largo de la línea el número de caracteres indicado por **n**. El signo - indica hacia la izquierda y el + hacia la derecha (por omisión se toma +).

Para ver un funcionamiento práctico basándonos en el ejemplo que hemos venido utilizando, imaginemos que deseamos modificar la línea 3 que contiene:

3: lugar

y en vez de "lugar" poner "local".

Las operaciones a realizar serán por este orden:

- Llevar el puntero al comienzo : **- b**
- Llevar puntero a línea 3 : **3 l**
- Llevar puntero a la letra l : **l c**
- Borrar "ugar" : **4 d**
- Añadir "ocal" : **i ocal**

*** Borrado de líneas completas**

Para borrar una o varias líneas consecutivas, ED dispone de la orden:

: ± n k

La letra **k** es la inicial de kill (suprimir) y **n** es el número de líneas a borrar, por delante o por detrás de la línea en curso (puntero) según el signo (- arriba, + abajo).

Por ejemplo para borrar la línea 3 completa, habrá que dar las siguientes órdenes:

l: * 3:	posicionar puntero en línea 3
3: * k	borrar línea 3
3: * b	ir al final del fichero
l: * # t	listar todo

Las órdenes de ED pueden concatenarse, de forma que el ejemplo de borrado anterior, también puede realizarse mediante:

l: * 3 k b # t

Cuando el valor de **n** es uno, puede omitirse.

El uso del comodín permite borrar todas las líneas anteriores o posteriores a la posición del puntero. La sintaxis será:

: ± # k

*** Conclusión de los trabajos**

Además de la orden ^C general de CP/M que produce una interrupción brusca de cualquier trabajo, ED contiene una orden propia de salida, mucho más aconsejable y útil que la anterior. Esta orden es **"q"**, inicial que **quit** (desistir).

Tras la introducción de esta orden, ED pide confirmación:

```
: *q
Q - (Y/N)?
```

Si se pulsa **Y**, las modificaciones son abandonadas. Si se pulsa **N**, la situación se mantiene inalterada, pasando a nivel de operador.

Una orden complementaria es **"o"** inicial de **original,** que deja el fichero en las condiciones originales, es decir, no se tienen en cuenta las modificaciones.

Al igual que la orden **"q",** tras la utilización de **"o",** ED pide confirmación:

```
: * o
O - (Y/N)?
```

El efecto es el mismo que con **"q"**, pero se mantiene el nivel de editor, en vez de pasar a nivel de operador.

Las órdenes del ED expuestas son sólo algunas de las utilizables con este programa, aunque estimamos suficientes para la creación y corrección de ficheros. Para mayor información véase la referencia citada al comienzo de este capítulo.

Resumen de órdenes del editor ED.COM

ED nombre	Carga el programa ED.COM y crea el fichero "nombre" si no existía, en caso contrario lo abre.
i	Insertar texto. Una o varias líneas.
#a	Transfiere el texto desde un fichero al buffer.
#t	Lista el fichero contenido en el buffer por pantalla e impresora, si ésta se encuentra activada por **^P.**

e	Transfiere el contenido del buffer al disco, almacenándolo en el fichero que se indicó.
n:	Inserta una línea en la posición indicada por el número **n.**
#a #t	Realiza simultáneamente las funciones de # a y #t.
± n d	Borrar el número de caracteres indicado por **n.**
± b	Puntero al principio o final del fichero.
± n l	Avanzar o retroceder **n** líneas.
± n c	Avanzar o retroceder **n** caracteres en una línea.
± n k	Borrar **n** líneas.
#	Comodín cuyo significado es "todo".
q	Salida a nivel de operador.
o	Salida manteniéndose a nivel de editor.

3. Utilización de los ficheros ED desde BASIC

* ED-BASIC

En el segundo apartado del capítulo 6, vimos cómo utilizar un fichero creado por EAMSWORD desde BASIC. Con los ficheros creados por ED también podemos proceder de una forma similar y mucho más sencilla, puesto que no existen caracteres de control ni información de otro tipo, sino únicamente la introducida en las líneas cuando se creó, con las modificaciones que se introdujeron durante su elaboración.

Para poder comprobarlo puede utilizarse la orden TYPE. Haga bajo CP/M:

```
A > TYPE PRUEBA.UNO
```

y obtendrá un listado del contenido del fichero línea a línea.

Por todo ello, un programa BASIC capaz de leer uno de estos ficheros puede ser el siguiente:

```
10 ' LECTURA DE FICHEROS ED
20 ' ......... ED1 ........
30 INPUT "NOMBRE DEL FICHERO";F$
40 INPUT "NUMERO DE LINEAS  ";N
50 DIM A$(N)
60 OPENIN F$
70 FOR K=1 TO N
80 INPUT #9,A$(K)
90 PRINT A$(K)
100 NEXT K
110 CLOSEIN
```

El programa permite leer todo el fichero cuando **N** es igual al número total de líneas del fichero; o parte de él, cuando **N** es menor.

Si el contenido del fichero fuesen datos numéricos, puede utilizarse una variable numérica dimensionada (línea 40), o bien una alfanumérica, realizando la transformación a numérica mediante la función VAL.

En caso de tener que leer todo el fichero -sin conocer de cuántas líneas se compone- debe sustituirse el bucle FOR por uno WHILE, según se indicó en el segundo apartado del capítulo 4: "Lectura con WHILE". (Ver programa FS3).

* BASIC-ED

El proceso puede invertirse, es decir, podemos crear un fichero desde BASIC y luego ser leído y modificado con el editor ED.COM, con lo cual el abanico de posibilidades es muy amplio.

Prepare un fichero BASIC con cualquiera de los variados programas incluidos en este libro y utilícelo con ED.COM y podrá comprobar las ventajas de manipulación que aporta el editor, cuyo manejo, si bien no resulta precisamente sencillo, un poco de práctica le hará familiarizarse con las órdenes más usuales.

La filosofía que se desprende después de trabajar con ficheros EAMSWORD y ED, es que, en principio, no hay inconvenientes en modificar y leer ficheros de muy diversa naturaleza desde distintos progamas, ya sean BASIC o Código Máquina.

Probablemente el lector posea otros tratamientos de textos y bases de datos y le gustaría encontrarles mayores aplicaciones. Siguiendo una metodología -como la expuesta en estos dos últimos capítulos- no le será difícil conseguirlo. No olvide que la orden TYPE de CP/M es una buena herramienta de ayuda.

Apéndice

CODIGO ASCII

SIM	DEC	HEX	BINARIO	COMENTARIO
NUL	0	00	00000000	NULO [C-@]
SQH	1	01	00000001	COMIENZO DE CABECERA [C-A]
STX	2	02	00000010	COMIENZO DE TEXTO [C-B]
ETX	3	03	00000011	FIN DE TEXTO [C-C]
EOT	4	04	00000100	FIN DE TRANSMISION [C-D]
ENQ	5	05	00000101	PETICION [C-E]
ACK	6	06	00000110	RECONOCIMIENTO [C-F]
BEL	7	07	00000111	ALARMA O CAMPANA [C-G]
BS	8	08	00001000	ESPACIO HACIA ATRAS [C-H]
HT	9	09	00001001	TABULACION HORIZONTAL [C-I]
LF	10	0A	00001010	ALIMENTACION DE LINEA [C-J]
VT	11	0B	00001011	TABULACION VERTICAL [C-K]
FF	12	0C	00001100	ALIMENTACION DE PAGINA [C-L]
CR	13	0D	00001101	RETORNO DEL CARRO [C-M]
SO	14	0E	00001110	DESPLAZAMIENTO EXTERNO [C-N]
SI	15	0F	00001111	DESPLAZAMIENTO INTERNO [C-O]
DLE	16	10	00010000	ESCAPE DEL ENLACE DE COMUNICACION DE DATOS [C-P]
DC1	17	11	00010001	CONTROL UNIDAD 1 [C-Q]
DC2	18	12	00010010	CONTROL UNIDAD 2 [C-R]
DC3	19	13	00010011	CONTROL UNIDAD 3 [C-S]
DC4	20	14	00010100	CONTROL UNIDAD 4 [C-T]
NAK	21	15	00010101	RECONOCIMIENTO NEGATIVO [C-U]
SYN	22	16	00010110	DOSECUPACION SINCRONA [C-V]
ETB	23	17	00010111	FIN DE TRANSMISION DE BLOQUE [C-W]
CAN	24	18	00011000	CANCELACION [C-X]
EM	25	19	00011001	FIN DE MEDIO [C-Y]
SUB	26	1A	00011010	SUSTITUCION [C-Z]
ESC	27	1B	00011011	ESCAPE [C-[]
FS	28	1C	00011100	SEPARADOR DE FICHERO [C-\]
GS	29	1D	00011101	SEPARADOR DE FRUPO [C-]]
RS	30	1E	00011110	SEPARADOR DE REGISTRO [C-^]
SS	31	1F	00011111	SEPARADOR DE UNIDAD [C-O]
SP	32	20	00100000	ESPACIO
!	33	21	00100001	SIMBOLOS
	34	22	00100010	
#	35	23	00100011	
$	36	24	00100100	
%	37	25	00100101	
&	38	26	00100110	
'	39	27	00100111	
(	40	28	00101000	
)	41	29	00101001	
*	42	2A	00101010	

SIM	DEC	HEX	BINARIO	COMENTARIO
+	43	2B	00101011	
,	44	2C	00101100	
-	45	2D	00101101	
.	46	2E	00101110	
/	47	2F	00101111	
0	48	30	00110000	NUMEROS
1	49	31	00110001	
2	50	32	00110010	
3	51	33	00110011	
4	52	34	00110100	
5	53	35	00110101	
6	54	36	00110110	
7	55	37	00110111	
8	56	38	00111000	
9	57	39	00111001	
:	58	3A	00111010	
;	59	3B	00111011	
<	60	3C	00111100	
=	61	3D	00111101	
>	62	3E	00111110	
?	63	3F	00111111	
@	64	40	01000000	
A	65	41	01000001	LETRAS MAYUSCULAS
B	66	42	01000010	
C	67	43	01000011	
D	68	44	01000100	
E	69	45	01000101	
F	70	46	01000110	
G	71	47	01000111	
H	72	48	01001000	
I	73	49	01001001	
J	74	4A	01001010	
K	75	4B	01001011	
L	76	4C	01001100	
M	77	4D	01001101	
N	78	4E	01001110	
O	79	4F	01001111	
P	80	50	01010000	
Q	81	51	01010001	
R	82	52	01010010	
S	83	53	01010011	
T	84	54	01010100	
U	85	55	01010101	

SIM	DEC	HEX	BINARIO	COMENTARIO
V	86	56	01010110	
W	87	57	01010111	
X	88	58	01011000	
Y	89	59	01011001	
Z	90	5A	01011010	
[	91	5B	01011011	SIMBOLOS
\	92	5C	01011100	
]	93	5D	01011101	
^	94	5E	01011110	
_	95	5F	01011111	
`	96	60	01100000	
a	97	61	01100001	LETRAS MINUSCULAS
b	98	62	01100010	
c	99	63	01100011	
d	100	64	01100100	
e	101	65	01100101	
f	102	66	01100110	
g	103	67	01100111	
h	104	68	01101000	
i	105	69	01101001	
j	106	6A	01101010	
k	107	6B	01101011	
l	108	6C	01101100	
m	109	6D	01101101	
n	110	6E	01101110	
o	111	6F	01101111	
p	112	70	01110000	
q	113	71	01110001	
r	114	72	01110010	
s	115	73	01110011	
t	116	74	01110100	
u	117	75	01110101	
v	118	76	01110110	
w	119	77	01110111	
x	120	78	01111000	
y	121	79	01111001	
z	122	7A	01111010	
{	123	7B	01111011	SIMBOLOS
|	124	7C	01111100	
}	125	7D	01111101	
	126	7E	01111110	

LISTA DE ERRORES

Error Nº	SIGNIFICADO
1	NEXT no esperado. No existe el FOR previo correspondiente.
2	Error de sintaxis. La línea no guarda las reglas de sintaxis del BASIC del Amstrad.
3	RETURN no esperado. No existe el GOSUB previo correspondiente.
4	DATA agotados. No hay más datos para leer con READ.
5	El argumento de la función no es válido.
6	Rebasamiento. El número es demasiado grande o pequeño.
7	Memoria llena.
8	No existe la línea que se indica.
9	El subíndice de la variable indexada está fuera del rango establecido en el DIM, o bien éste no existe y el subíndice es mayor de 10.
10	La matriz ya ha sido dimensionada.
11	División por cero.
12	Comando directo no válido.
13	Incongruencia de tipos (números y textos).
14	No hay más sitio para textos.
15	Cadena demasiado larga (más de 256 caracteres).
16	Cadena demasiado compleja.
17	No se puede continuar con CONT después de una interrupción.
18	Función desconocida. Hay un FN sin el DEF FN correspondiente.
19	Orden RESUME perdida. Se ha llegado al final del programa mientras se ejecutaba un ON ERROR.
20	RESUME no esperado. No existe el ON ERROR previo correspondiente.
21	Encontrado un comando directo (sin número de línea).
22	Operando perdido, por operación incompleta.
23	Línea demasiado larga.
24	Se ha encontrado leyendo un fichero la señal de final del mismo.
25	Tipo de fichero erróneo.
26	NEXT perdido. Hay un FOR sin su correspondiente NEXT.
27	El fichero ya estaba abierto.
29	WEND perdido. Hay un WHILE sin su WEND correspondiente.
30	WEND inesperado. Se ha encontrado un WEND sin su WHILE correspondiente.

ERRORES DE DISCO

Error Nº	SIGNIFICADO
31	File not open
32	Broken in

Valores de DERR corresponndientes

Valor DERR	Tipo de error
0 ó 22	Se ha pulsado ESC.
142	Situación incorrecta del canal del disco.
143	Final del fichero (fin físico)
144	Error en la orden o nombre incorrecto del fichero
145	El fichero indicado ya existe en el disco
146	El fichero indicado no existe en el disco
147	Directorio agotado
148	Disco lleno
149	Se ha removido el disco cuando había ficheros abiertos en él.
150	Fichero de sólo lectura.
154	No hay más datos en el fichero (final lógico).

Otros libros AMSTRAD publicados por RA-MA:

OTROS LIBROS PARA USUARIOS AMSTRAD PUBLICADOS POR RA-MA
=======

-AMSTRAD CPC-464 PROGRAMACION AVANZADA: Guia del Usuario.
M. Harrison.
Abril 1985 - rústica - 153 páginas -

-TECNICAS DE PROGRAMACION AVANZADA CON AMSTRAD.
K. Hook.
Octubre 1985 - rústica - 161 páginas.

-AMSTRAD CPC-464/664/6128 PROGRAMACION ESTRUCTURADA.
S. Raven.
Diciembre 1985 - rústica - 174 páginas.

-APRENDE LOGO CON AMSTRAD: Ficheros en Castellano.
Spen, S.A.
Mayo 1986 - libro de 104 páginas + disco traductor del LOGO Ingles.

-DOMINE EL CODIGO MAQUINA EN SU AMSTRAD CPC-6128/664/464.
C. Gifford, y S. Vincent.
Junio 1986 - rústica - 228 páginas.

-RUTINAS EN CODIGO MAQUINA PARA SU AMSTRAD.
C. Gifford, y S. Vincent.
Mayo 1986 - rústica - 92 páginas.

-AMSTRAD CPC-HARDWARE: Conocimiento y Ampliación.
A.Trevennor.
Noviembre 1986 - rústica - 350 páginas.
De forma fácil y amena el libro le detalla toda la descripción del "Hardware" interno.

-GRAFICOS AVANZADOS CON AMSTRAD.
R. Ransom.
Diciembre 1986 - rústica - 230 páginas.
Con amplio detalle de programas y rutinas sobre técnicas de gráficos, este libro cubre desde el dibujo de figuras sencillas, hasta aplicaciones tan avanzadas como la representación de moléculas.

-CP/M GUIA DEL PROGRAMADOR. CP/M plus, 2.2, y 1.4.
A.Clarke, J:M.Eaton, D. Powys-Lybbe.
Enero 1987 - rústica - 350 páginas.
El libro ha merecido la aprobación de Amstrad en el Reino Unido, y ha sido recomendada la compra a sus usuarios.

-EL DOMINIO DEL AMSTRAD PCW-8256/8512.
J.M. Hughes.

LOS FICHEROS EN EL AMSTRAD CPC 464, CPC 664 y CPC 6128

Este libro dirigido a los usuarios de los CPC's de Amstrad, trata desde temas de conocimiento general, como matrices y ordenación aplicables a cualquier modelo de ordenador, a ficheros específicos de los citados modelos, de acceso directo o secuencial.

Además se incluyen numerosas rutinas, ejemplos y programas de utilidad —EAMSWORD , ED.COM (CP/M) etc.— que manejándolos desde Basic se pretende darles una mayor utilidad.

RA — MA
Carretera de Canillas, 144.
28043 Madrid.

ISBN 84-86381-23-1

LOS FICHEROS EN EL AMSTRAD CPC 464, CPC 664 Y CPC 6128.

VICTOR J. CAMPO

ra-ma

AMSTRAD CPC

MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA

https://acpc.me/